STREAMING SIMULTANEOUS SPEECH TRANSLATION WITH AUGMENTED MEMORY TRANSFORMER

Xutai Ma[†] *Yongqiang Wang*[⋆] *Mohammad Javad Dousti*[⋆] *Philipp Koehn*[†⋆] *Juan Pino*[⋆]

[⋆] Facebook
[†] Johns Hopkins University

ABSTRACT

Transformer-based models have achieved state-of-the-art performance on speech translation tasks. However, the model architecture is not efficient enough for streaming scenarios since self-attention is computed over an entire input sequence and the computational cost grows quadratically with the length of the input sequence. Nevertheless, most of the previous work on simultaneous speech translation, the task of generating translations from partial audio input, ignores the time spent in generating the translation when analyzing the latency. With this assumption, a system may have good latency quality trade-offs but be inapplicable in real-time scenarios. In this paper, we focus on the task of *streaming simultaneous speech translation*, where the systems are not only capable of translating with partial input but are also able to handle very long or continuous input. We propose an end-to-end transformer-based sequence-to-sequence model, equipped with an *augmented memory transformer encoder*, which has shown great success on the streaming automatic speech recognition task with hybrid or transducer-based models. We conduct an empirical evaluation of the proposed model on segment, context and memory sizes and we compare our approach to a transformer with a unidirectional mask. [1]

***Index Terms*—** End-to-end speech translation, Simultaneous speech translation, Streaming speech translation

1. INTRODUCTION

Streaming speech translation targets low latency scenarios such as simultaneous interpretation. Unlike the streaming automatic speech recognition (ASR) task where input and output are monotonically aligned, translation needs a larger future context due to reordering. Generally, simultaneous translation models start to translate with partial input, and then alternate between generating predictions and consuming additional input. While most previous work on simultaneous translation focus on text input, [1, 2, 3, 4, 5] the end-to-end approach for simultaneous speech translation has also very recently attracted interest from the community [6, 7] due to potentially lower latency compared with cascade models [8, 9, 10, 11]. However, most studies tend to focus on an ideal setup, where the computation time to generate the translation is neglected. This assumption may be reasonable for text to text but not for speech to text translation since the latter has much longer input sequences. A simultaneous speech translation model may have the ability to generate translations with partial input but may not be useful for real-time applications because of slow computation in generating output tokens.

While the majority of previous work on streaming speech to text tasks has focused on ASR, most prior work on ASR is not directly applicable to translation. Encoder-only or transducer structures are widely implemented, since ASR assumes the output is monotonically aligned to the input. In order to achieve an efficient streaming simultaneous speech translation model, we combine streaming ASR and simultaneous translation techniques and introduce an end-to-end transformer-based speech translation model with an augmented memory encoder [12].

The augmented memory encoder has shown considerable improvements on latency with little sacrifice on quality with hybrid or transducer-based models on the ASR task. It incrementally encodes fixed-length sub-sentence level segments and stores the history information with a memory bank, which summarizes each segment. The self-attention is only performed on the current segment and memory banks. A decoder with simultaneous policies is then introduced on top of the encoder. We apply the proposed model to the simultaneous speech translation task on the MuST-C dataset [13].

This paper is organized as follows. We first define the evaluation method for simultaneous speech translation. We then introduce the model based on the augmented memory transformer. Finally, we conduct a series of experiments to demonstrate the effectiveness of the proposed approach.

2. EVALUATION

This paper focuses on streaming *and* simultaneous speech translation, which features two additional capabilities compared to a traditional offline model. The first one is the ef-

[1] This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

ficient computation needed to handle streaming input, and the second is the ability to start a translation with partial input, then dynamically generate additional tokens or read additional input. Both factors need to be considered for evaluation.

A simultaneous system is evaluated with respect to quality, usually with BLEU, and latency. Latency is evaluated with computation-aware and non computation-aware Average Lagging (AL) [7, 3]. Denote the input sequence as $\mathbf{X} = [\mathbf{x}_1, ...]$, where each element is a feature vector extracted from a sliding window of size T and the translation of the system $\mathbf{Y} = [y_1, ...]$ and a reference translation $\mathbf{Y}^*$. Non computation-aware (NCA) latency is defined as follows

$$\text{AL} = \frac{1}{\tau(|\mathbf{X}|)} \sum_{i=1}^{\tau(\mathbf{X})} d(y_i) - \frac{|\mathbf{X}|}{|\mathbf{Y}^*|} \cdot T \cdot (i-1) \tag{1}$$

where $\tau(|X|)$ is the index of the first target token when the system has read the entire source input, and $d(y_i)$ is the duration of the speech that has been read when generating word y_i. Additionally, a computation-aware (CA) version of AL is also considered, by replace $d(y_i)$ with the time needed to generate y_i.

3. MODEL

The proposed streaming speech translation model, illustrated in Fig. 1, consists of two components, an augmented memory encoder and a simultaneous decoder. The encoder incrementally and efficiently encodes streaming input, while the decoder starts translation with partial input, then interleaves reading new input and predicting target token under the guidance of a simultaneous translation policy.

3.1. Augmented Memory Encoder

The self-attention module in the original transformer model [14] attends to the entire input sequence, which precludes streaming capability. Denote $\mathbf{H} = [\mathbf{h}_1, ..]$ the input of a certain encoder layer. Each self-attention projects the input into query, key and value.

$$\mathbf{Q} = W_q\mathbf{H}, \mathbf{K} = W_k\mathbf{H}, \mathbf{V} = W_v\mathbf{H} \tag{2}$$

At each position j, a weight is calculated as follows

$$\alpha_{jj'} = \frac{\exp(\beta \cdot \mathbf{Q}_j^T \mathbf{K}_{j'})}{\sum_k \exp(\beta \cdot \mathbf{Q}_j^T \mathbf{K}_k)} \tag{3}$$

The self-attention at position j can then be calculated as

$$\mathbf{Z}_j = \sum_{j'} \alpha_{jj'} \mathbf{V}_{j'} \tag{4}$$

The calculation of self-attention make it inefficient for streaming applications. [12] proposes an augmented memory transformer encoder to address this issue. Instead of attending to entire input sequence $\mathbf{X}$, the self-attentions are applied on a sequence of sub-utterance level segments $\mathbf{S} = [\mathbf{s}_1, ...]$. A segment $\mathbf{s}_n$, which contains a span of input features, consists of three parts: left context $\mathbf{l}_n$ of size L, main context $\mathbf{c}_n$ of size C and right context $\mathbf{r}_n$ of size R. Each segment overlaps with adjacent segments — the overlap between current and previous segment is $\mathbf{l}_n$, and between current and the next segment is $\mathbf{r}_n$. Self-attention is computed at the segment level, which reduces the amount of computation. The new query, key and value for each segment are

$$\mathbf{q}_n = \mathbf{W}_\mathbf{q}(\mathbf{l}_n, \mathbf{c}_n, \mathbf{r}_n, \sigma_n) \tag{5}$$

$$\mathbf{k}_n = \mathbf{W}_\mathbf{k}(\mathbf{M}_{n-N:n-1}, \mathbf{l}_n, \mathbf{c}_n, \mathbf{r}_n) \tag{6}$$

$$\mathbf{v}_n = \mathbf{W}_\mathbf{v}(\mathbf{M}_{n-N:n-1}, \mathbf{l}_n, \mathbf{c}_n, \mathbf{r}_n) \tag{7}$$

Where $\sigma_n = \sum_{\mathbf{x}_k \in \mathbf{s}_n} \mathbf{x}_k$ is a summarization of the segment $\mathbf{s}_n$, and $\mathbf{M}_{n-N:n} = [\mathbf{m}_{n-N}, ..., \mathbf{m}_{n-1}]$ are the memory banks. Each memory bank is calculated as follows:

$$\mathbf{m}_n = \sum_{j'} \alpha_{-1,j'} (\mathbf{v}_n)_{j'} \tag{8}$$

which is introduced to represent history information. A hyperparameter N controls how many memory banks are retained. The self-attention is then calculated as follows:

$$\alpha_{jj'} = \frac{\exp(\beta \cdot (\mathbf{q}_n^T)_j (\mathbf{k}_n)_{j'})}{\sum_k \exp(\beta \cdot (\mathbf{q}_n^T)_j (\mathbf{k}_n)_k)} \tag{9}$$

$$(\mathbf{z}_n)_j = \sum_{j'} \alpha_{j,j'} (\mathbf{v}_n)_{j'} \tag{10}$$

$$\text{where } N + L < j \leq N + L + C \tag{11}$$

Then only the central encoder states are kept and a the concatenation of the segment states $\mathbf{Z} = [\mathbf{z}_1, ...]$ is passed to decoder. Because of the left and right contexts, an arbitrary encoder can run on the segments without boundary mismatch. In this paper, we adapt the encoder of the convtransformer architecture [15]. The encoder first consists of convolutional layers with stride 2 that subsample the input. Full self-attention layers can then be calculated.

3.2. Simultaneous Decoder

A simultaneous decoder starts translation with partial input based on a policy. Simultaneous policies decide whether the model should read new inputs or generate a new prediction at a given time. However, different from text translation, our preliminary experiments show that for simultaneous speech translation, encoder states are too granular for policy learning. Thus, we adopt the idea of pre-decision [7] for better efficiency by making simultaneous read and write decision on chunks of encoder states. Here, the simpler fixed pre-decision strategy is used where the decision is made every fixed number of encoder states. Denote the sequence of chunks are $\mathbf{W} = [\mathbf{w}_1, ...]$ and the start and end encoder

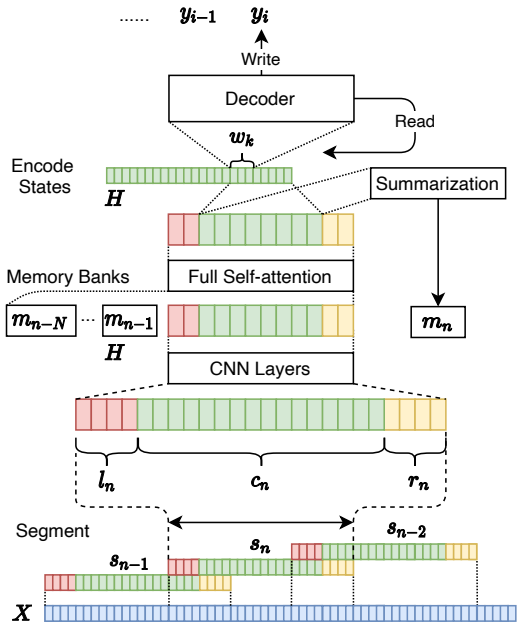

Fig. 1: Architecture of streaming transformer model with an augmented memory encoder.

state index of $\mathbf{w}_k$ is $W_s(k), W_e(k)$. Denote the prediction of model $\mathbf{Y} = [y_1, ...]$, the general decoding algorithm of a simultaneous policy $\mathcal{P}$ with augmented memory transformer is described in Algorithm 1. In theory, Algorithm 1 supports arbitrary simultaneous translation policies. In this paper, for simplicity, wait-k [3] is used. It waits for k source tokens and then operating then reading and writing alternatively. Notice that our method is compatible with an arbitrary simultaneous translation policy.

Note that the decoder self-attention still has access to all previous decoder hidden states; in order to preserve streaming capability for the decoder, decoder states are reset every time an end-of-sentence token is predicted. The augmented memory is not introduced in the decoder because the target sequence is dramatically smaller than the source speech sequence. The decoder can still predict a token in a negligible time compared with encoding source with the input becoming longer.

4. EXPERIMENTS

Experiments were conducted on the English-German MuST-C dataset [13]. The training data consists of 408 hours of speech and 234k sentences of text. We use Kaldi [16] to extract 80 dimensional log-mel filter bank features. The features are computed with a $25ms$ window size and a $10ms$ window shift and normalized with global cepstral mean and variance. Text is tokenized with a SentencePiece[2] 10k unigram vocabulary. Translation quality is evaluated with case-sensitive detokenized BLEU with SACREBLEU[3]. The latency is evaluated by Average Lagging [3, 7], with the SimulEval toolkit[4].

The speech translation model is based on the convtransformer architecture [15]. It first contains two convolutional layers with subsampling ratio of 4. Both encoder and decoder have a hidden size of 256 and 4 attention heads. There are 12 encoder layers and 6 decoder layers. The model is trained with label smoothed (0.1) cross entropy. We use the Adam optimizer [17], with a learning rate of 0.0001 and an inverse square root schedule.

We use a simplified version of [6] as our baseline model. A unidirectional mask is introduced to prevent the encoder from looking into future information. For baseline models, we follow common practice for simultaneous text translation where the entire encoder is updated once there is new input. Instead of a multi-task setting and a decision making process depending on word boundaries obtained from the auxiliary ASR task, we make decisions on a fixed size chunk of encoder states, following [7]. Our choice is motivated by the fact that in [7], a fixed chunk size gave similar quality-latency trade-offs as word boundaries.

All transformer-based speech translation models are first pre-trained on the ASR task, in order to initialize the encoder. Each experiment is run on 8 Tesla V100 GPUs with 32 GB memory. The code is developed based on Fairseq[5], and it will be published upon acceptance. All scores are reported on the dev set.

Algorithm 1 Chunk-based simultaneous policy with an augmented memory encoder

Require: Chunk-based simultaneous policy $\mathcal{P}$
Require: Streaming input $\mathbf{X}$. Memory banks $\mathbf{M}$. Prediction $\mathbf{Y}$
Require: Maximum memory size N. Decision chunk size W
Require: Central context size C. Encoder pooling ratio R
Require: $i = 1, n = 1, k = 1$.
Require: $W_e(1) = 1, y_0 = \text{BOS}$

while $y_{i-1} \neq$ EndOfTranslation **do**
  if $W_e(k) + W > n \cdot C \cdot R$ **then**
    $\mathbf{z}_n, \mathbf{m}_n = \text{Encoder}(\mathbf{s}_n, \mathbf{M}_{n-N:n-1})$
    $\mathbf{Z} = [\mathbf{Z}, \mathbf{z_n}], \mathbf{M} = [\mathbf{M}, \mathbf{m_n}]$
    $n = n + 1$ # Read a new segment of input features
  $\mathbf{w}_k = \text{Summarize}(\mathbf{Z}_{W_s(k):W_e(k)})$
  $p_{ik} = \mathcal{P}([\mathbf{Y}_{1:i-1}], \mathbf{w}_k)$
  if $p_{ik} > 0.5$ **then**
    $y_i = \text{Decoder}([\mathbf{Y}_{1:i-1}], \mathbf{Z})$
    $i = i + 1$ # Predict a target token
  else
    $W_s(k+1) = W_e(k) + 1$
    $k = k + 1$ # Move to the next chunk of encoder states

[2] https://github.com/google/sentencepiece

[3] https://github.com/mjpost/sacrebleu

[4] https://github.com/facebookresearch/SimulEval

[5] https://github.com/pytorch/fairseq

5. RESULTS

We first analyze the effect of the segment and context sizes, and use the resulting optimal settings for further analysis on the maximum number of memory banks and for comparison with the baseline.

5.1. Effect of Segment and Context Size

We analyze the effect of different segment, left and right context sizes. For all experiments, we use the wait-k ($k = 1, 3, 5, 7$) policy on a chunk of 8 encoder states [7]. The latency-quality trade-offs with different sizes are shown in Figure 2. We first observe that, increasing the left and right context size will improve the quality with very small trade-off on latency, for instance, from curve "S64 L16 R16" to "S64 L32 R32". This indicates that context of both sides can alleviate the boundary effect. We also notice that when we reduce the segment size from 64 to 32, the BLEU score decreases dramatically. Similar observations are made in [12] but the ASR models are more robust to decreasing the segment and context sizes. We hypothesize that reordering in translation makes the model more sensitive to these sizes.

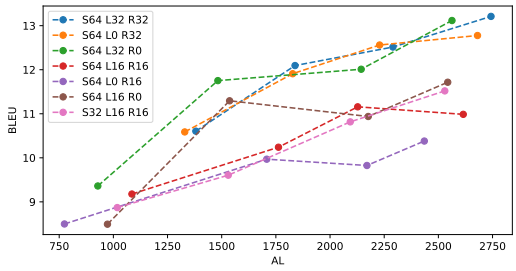

Fig. 2: Effect of segment, left and right context size. Each curve represents wait-k, $k = 1, 3, 5, 7$ policies. The size is measured on a frame of 10 ms. "S$\{x\}$ L$\{y\}$ R$\{z\}$" means a encoder with segment size x, left size y and right size x.

5.2. Number of Memory Banks

In streaming translation, the input is theoretically infinite. In order to prevent memory explosion, we explore the effect of reducing the number of the memory banks. Fig. 3 shows the effect of different numbers of memory banks. We can see that the model is very robust to the size of the memory banks. Similar to [12], when the maximum number of memory banks is large, for instance, larger than 3, there is little or no performance drop. However, we still observe a drop in performance with a maximum number of one memory bank. Finally, we found that training with different maximum numbers of memory banks was necessary as limiting the number of memory banks only at inference time degraded performance.

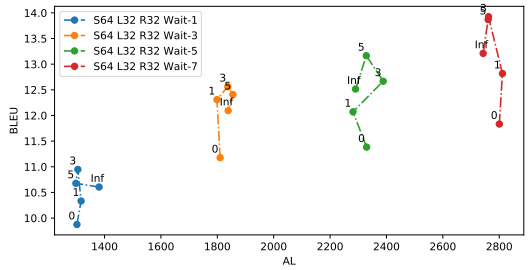

Fig. 3: Effect of the maximum number of memory banks. Each curve represents one policy. The number on top of the nodes indicates the number of memory banks being kept.

5.3. Comparison with Baseline

In Fig. 4, we compared our model with the baseline model described in Section 4. The proposed model achieves better quality with an increase in computation aware and non computation aware latency. The baseline achieves competitive latency because it only updates encoder states every 8 steps. However, there may be instances where recomputing encoder states every step may be needed, for example in the case of a flexible pre-decision module or when a the model includes a boundary detector [6]. We can see in Fig. 4 that the computation aware AL for the baseline increases substantially with a chunk of size 1.

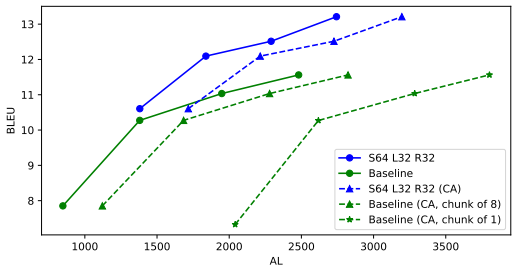

Fig. 4: Comparison with baseline model. CA indicates measured by computation aware latency. Chunk of x means that the encoder states are updated every x steps.

6. CONCLUSION

In this paper, we tackle the real-life application of streaming simultaneous speech translation. We propose a transformer-based model, equipped with an augmented memory, in order to handle long or streaming input. We study the effect of segment and context sizes, and the maximum number of memory banks. We show that our model has better quality with an acceptable latency increase compared with a transformer with unidirectional mask baseline and presents better quality-latency trade-offs than that baseline where encoder states are recomputed at every step.

7. REFERENCES

[1] Kyunghyun Cho and Masha Esipova, "Can neural machine translation do simultaneous translation?," *arXiv preprint arXiv:1606.02012*, 2016.

[2] Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor OK Li, "Learning to translate in real-time with neural machine translation," in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*. Association for Computational Linguistics (ACL), 2017, pp. 1053–1062.

[3] Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang, "STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, July 2019, pp. 3025–3036, Association for Computational Linguistics.

[4] Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel, "Monotonic infinite lookback attention for simultaneous machine translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, July 2019, pp. 1313–1323, Association for Computational Linguistics.

[5] Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu, "Monotonic multihead attention," in *International Conference on Learning Representations*, 2019.

[6] Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, QIN Tao, Zhou Zhao, and Tie-Yan Liu, "Simulspeech: End-to-end simultaneous speech to text translation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 3787–3796.

[7] Xutai Ma, Juan Pino, James Cross, Liezl Puzon, and Jiatao Gu, "SimulMT to SimulST: Adapting Simultaneous Text Translation to End-to-End Simultaneous Speech Translation," in *Proceedings of 2020 Asia-Pacific Chapter of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, 2020.

[8] Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura, "Optimizing segmentation strategies for simultaneous speech translation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, pp. 551–556.

[9] Christian Fügen, Alex Waibel, and Muntsin Kolss, "Simultaneous translation of lectures and speeches," *Machine translation*, vol. 21, no. 4, pp. 209–252, 2007.

[10] Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura, "Simple, lexicalized choice of translation timing for simultaneous speech translation.," in *INTERSPEECH*, 2013, pp. 3487–3491.

[11] Markus Müller, Thai Son Nguyen, Jan Niehues, Eunah Cho, Bastian Krüger, Thanh-Le Ha, Kevin Kilgour, Matthias Sperber, Mohammed Mediani, Sebastian Stüker, et al., "Lecture translator-speech translation framework for simultaneous lecture translation," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 2016, pp. 82–86.

[12] Chunyang Wu, Yongqiang Wang, Yangyang Shi, Ching-Feng Yeh, and Frank Zhang, "Streaming transformer-based acoustic models using self-attention with augmented memory," *Interspeech*, 2020.

[13] Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi, "MuST-C: a Multilingual Speech Translation Corpus," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, June 2019, pp. 2012–2017, Association for Computational Linguistics.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe, "ESPnet-ST: All-in-one speech translation toolkit," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Online, July 2020, pp. 302–311, Association for Computational Linguistics.

[16] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number CONF.

[17] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.