Direct Simultaneous Speech-to-Text Translation Assisted by Synchronized Streaming ASR *

Junkun Chen [1] **Mingbo Ma** [2] **Renjie Zheng** [2] **Liang Huang** [1,2]
[1]Oregon State University, Corvallis, OR, USA
[2]Baidu Research, Sunnyvale, CA, USA
chenjun2@oregonstate.edu, mingboma@baidu.com

Abstract

Simultaneous speech-to-text translation is widely useful in many scenarios. The conventional cascaded approach uses a pipeline of streaming ASR followed by simultaneous MT, but suffers from error propagation and extra latency. To alleviate these issues, recent efforts attempt to directly translate the source speech into target text simultaneously, but this is much harder due to the combination of two separate tasks. We instead propose a new paradigm with the advantages of both cascaded and end-to-end approaches. The key idea is to use two separate, but synchronized, decoders on streaming ASR and direct speech-to-text translation (ST), respectively, and the intermediate results of ASR guide the decoding policy of (but is not fed as input to) ST. During training time, we use multitask learning to jointly learn these two tasks with a shared encoder. En-to-De and En-to-Es experiments on the MuST-C dataset demonstrate that our proposed technique achieves substantially better translation quality at similar levels of latency.

1 Introduction

Simultaneous speech-to-text translation incrementally translates source-language speech into target-language text, and is widely useful in many cross-lingual communication scenarios such as international travels and multinational conferences. The conventional approach to this problem is a cascaded one (Arivazhagan et al., 2020; Xiong et al., 2019; Zheng et al., 2020b), involving a pipeline of two steps. First, the streaming automatic speech recognition (ASR) module transcribes the input speech on the fly (Moritz et al., 2020; Wang et al., 2020), and then a simultaneous text-to-text translation module translates the partial transcription into target-language text (Oda et al., 2014; Dalvi et al., 2018; Ma et al., 2019; Zheng et al., 2019a,b, 2020a; Arivazhagan et al., 2019).

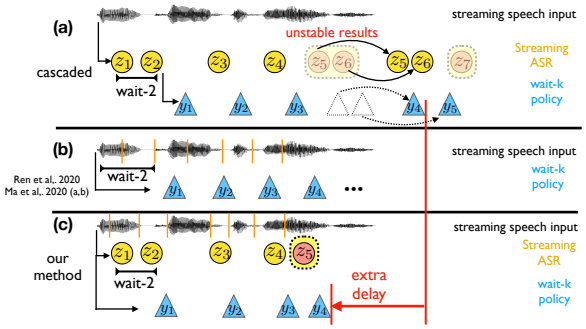

Figure 1: Comparison between (a) cascaded pipeline, (b) direct simultaneous ST, and (c) our ASR-assisted simultaneous ST. In (a), streaming ASR keeps revising some tail words for better accuracy, but causing extra delays to MT. Method (b) directly translates source speech without using ASR. Our work (c) uses the intermediate results of the streaming ASR module to guide the decoding policy of (but not feed as input to) the speech translation module. Extra delays between ASR and MT are reduced in direct translation systems (b–c).

However, the cascaded approach inevitably suffers from two limitations: (a) **error propagation**, where streaming ASR's mistakes confuse the translation module (which are trained on clean text), and this problem worsens with noisy environments and accented speech; and (b) **extra latency**, where the translation module has to wait until streaming ASR's output stabilizes, as ASR by default can repeatedly revise its output (see Fig. 1).

To overcome the above issues, some recent efforts (Ren et al., 2020; Ma et al., 2020b,a) attempt to directly translate the source speech into target text simultaneously by adapting text-based wait-k strategy (Ma et al., 2019). However, unlike simultaneous translation whose input is already segmented into words or subwords, in speech translation, the key challenge is to figure out the number of valid tokens within a given source speech segment in or-

* See our translation examples and demos at https://littlechencc.github.io/SimulST-demo/simulST-demo.html.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4618–4624
August 1–6, 2021.

der to apply the wait-k policy. Ma et al. (2020b,a) simply assume a fixed number of words within a certain number of speech frames, which does not consider various aspects of speech such as different speech rate, duration, pauses and silences, all of which are common in realistic speech. Ren et al. (2020) design an extra Connectionist Temporal Classification (CTC)-based speech segmenter to detect the word boundaries in speech. However, the CTC-based segmenter inherits the same shortcoming of CTC, which only makes local predictions, thus limiting its segmentation accuracy. On the other hand, to alleviate the error propagation, Ren et al. (2020) employ several different knowledge distillation techniques to learn the attentions of ASR and MT jointly. These knowledge distillation techniques are complicated to train and it is an indirect solution for the error propagation problem.

We instead present a simple but effective solution (see Fig. 2) by employing two separate, but synchronized, decoders, one for streaming ASR and the other for End-to-End Speech-to-text Translation (E2E-ST). Our key idea is to use the intermediate results of streaming ASR to guide the decoding policy of, but not feed as input to, the E2E-ST decoder. We look at the beam of streaming ASR to decide the number of tokens within the given source speech segment. Then it is straightforward for the E2E-ST decoder to apply the wait-k policy and decide whether to commit a target word or to wait for more speech frames. During training time, we jointly train ASR and E2E-ST tasks with a shared speech encoder in a multi-task learning (MTL) fashion to further improve the translation accuracy. We also note that having streaming ASR as an auxiliary output is extremely useful in real application scenarios where the user often wants to see both the transcription and the translation. En-to-De and En-to-Es experiments on the MuST-C dataset demonstrate that our proposed technique achieves substantially better translation quality at similar level of latency.

2 Preliminaries

We formalize full-sentence tasks (ASR, MT and ST) using the sequence-to-sequence framework, and the streaming tasks (simultaneous MT and streaming ASR) using the test-time wait-k method.

Full-Sentence Tasks: ASR, NMT and ST The encoder first encodes the entire source input into a sequence of hidden states; in NMT, the input is a sequence of words, $\mathbf{x} = (x_1, x_2, ..., x_m)$, while in ASR and ST, we use $\mathbf{s}$ to denote the input speech frames. A decoder sequentially predicts target language tokens $\mathbf{y} = (y_1, y_2, ..., y_n)$ in NMT and ST or transcription $\mathbf{z}$ in ASR, conditioned on all encoder hidden states and previously committed tokens. For example, the NMT model and its parameters $\boldsymbol{\theta}^{\text{MT}}_{\text{full}}$ are defined as:

$$p_{\text{full}}(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}^{\text{MT}}_{\text{full}}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid \mathbf{x}, \mathbf{y}_{<t}; \boldsymbol{\theta}^{\text{MT}}_{\text{full}})$$

$$\hat{\boldsymbol{\theta}}^{\text{MT}}_{\text{full}} = \mathbf{argmax}_{\boldsymbol{\theta}^{\text{MT}}_{\text{full}}} \prod_{(\mathbf{x},\mathbf{y}^*)\in D} p_{\text{full}}(\mathbf{y}^* \mid \mathbf{x}; \boldsymbol{\theta}^{\text{MT}}_{\text{full}})$$

Similarly, we can obtain the definitions for ASR ($p_{\text{full}}(\mathbf{z} \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ASR}}_{\text{full}})$) and ST ($p_{\text{full}}(\mathbf{y} \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ST}}_{\text{full}})$). Our model was learned from scratch in this work, but it can be improved with pre-training methods (Zheng et al., 2021; Chen et al., 2020).

Simultaneous MT and Streaming ASR In streaming decoding scenarios, we have to predict target tokens conditioned on the partial source input that is available. For example, the **test-time wait-k** method of Ma et al. (2019) predicts each target token y_t after reading source tokens $\mathbf{x}_{\leq t+k}$ using a full-sentence NMT model:

$$\hat{y}_t = \mathbf{argmax}_{y_t}\, p_{\text{wait-}k}(y_t \mid \mathbf{x}_{\leq t+k}, \hat{\mathbf{y}}_{<t}; \hat{\boldsymbol{\theta}}^{\text{MT}}_{\text{full}}) \quad (1)$$

Intuitively speaking, wait-k only commits a new target word on receiving each new source word after an initial k source words waiting. Similarly, in the case of streaming ASR, we could define $\hat{z}_t$ with growing speech chunks $\bar{\mathbf{s}}_i$ that are fed gradually.

3 Direct Simultaneous Translation with Synchronized Streaming ASR

In text-to-text simultaneous translation, the input stream is already segmented. However, when we deal with speech frames as source inputs, it is not easy to determine the number of valid tokens within certain speech segments. Therefore, to better guide the translation policy, it is essential to detect the number of valid tokens accurately within low latency. Different from the sophisticated design of speech segmenter in Ren et al. (2020), we propose a simple but effective method by using a synchronized streaming ASR and using its beam to determine the number of words within certain speech segments. Note that we only use streaming ASR for source word counting, but the translation decoder does not condition on any of ASR's output.

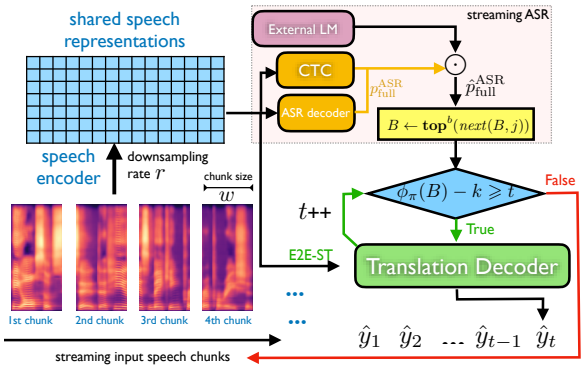

Figure 2: Decoding for synchronized streaming ASR and E2E-ST. Speech signals are fed into the encoder chunk by chunk. For each new-coming speech chunk, we look at the current streaming ASR beam (B) to decide the translation policy. See details in Algorithm 1.

3.1 Streaming ASR-Guided Simultaneous ST

As shown in Fig. 2, at inference time, the speech signals are fed into the ST encoder by a series of fixed-size chunks $\bar{\mathbf{s}}_{[1:i]} = [\bar{\mathbf{s}}_1, ..., \bar{\mathbf{s}}_i]$, where $w = |\bar{\mathbf{s}}_i|$ can be chosen from 32, 48 and 64 frames of spectrogram. As a result of the CNN encoder, there is down sampling rate r (e.g., we use $r = 4$), from spectrogram to encoder hidden states. For example, when we receive a chunk of 32 frames, the encoder will generate 8 more hidden states. In conventional streaming ASR, the number of steps of beam search is the same as the number of hidden states.

We denote B_j to be the beam at time step j, which is an ordered list of size of b, and it expands to the next beam B_{j+1} with the same size:

$$
\begin{aligned}
B_0 &= [\langle \text{<s>}, \hat{p}^{\text{ASR}}_{\text{full}}(\text{<s>} \mid \bar{\mathbf{s}}_0; \boldsymbol{\theta}) \rangle] \\
B_j &= \mathbf{top}^b(next(B_{j-1}, j)) \\
next(B, j) &= \{ \langle \mathbf{z} \circ z_j,\ p \cdot \hat{p}^{\text{ASR}}_{\text{full}}(z_j \mid \bar{\mathbf{s}}_{\leq \tau(j)}, \mathbf{z}; \boldsymbol{\theta}) \rangle \mid \\
&\qquad \langle \mathbf{z}, p \rangle \in B, z_j \in V \}
\end{aligned}
$$

where $\mathbf{top}^b(\cdot)$ returns the top b candidates, and $next(B, j)$ expands the candidates from the previous step to the next step. Each candidate is a pair $\langle \mathbf{z}, p \rangle$, where $\mathbf{z}$ is the current prefix and p is the accumulated probability from joint score between an external language model, CTC and ASR probabilities, $\hat{p}^{\text{ASR}}_{\text{full}}$. We denote the number of observable speech chunks at j step as $\tau(j) = \lceil j * r/w \rceil$. And vice versa, for each new speech chunk, ASR beam search will advance for w/r steps.

Note CTC often commits empty tokens ϵ due to empty speech frames, and the lengths of different hypotheses within beam of streaming ASR are quite different from each other. To take every hypothesis into consideration, we design two policies to decide the number of valid tokens.

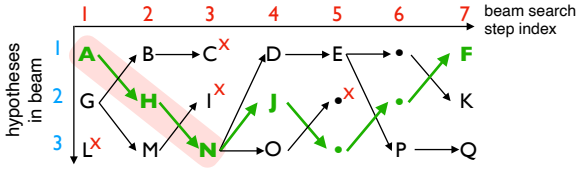

Figure 3: An example of streaming ASR beam search with beam size 3. LCP is shaded in red ($\phi_{\text{LCP}}(B_7) = 3$); SH is highlighted in bold ($\phi_{\text{SH}}(B_7) = 5$). We use • to represent empty outputs in some steps caused by CTC.

Algorithm 1 Streaming ASR-guided Simultaneous ST

Input: speech chunks $\bar{\mathbf{s}}_{[1:T]}$; k; $\phi_\pi(B_j)$; streaming decoding models: $p^{\text{ST}}_{\text{full}}$ and $\hat{p}^{\text{ASR}}_{\text{full}}$
Initialize: ASR and ST indices: $j = t = 0$; $B = B_0$
for $i = 1 \sim T$ **do** ▷ feed speech chunks
  repeat w/r steps ▷ do ASR beam search w/r steps
    $B \leftarrow \mathbf{top}^b(next(B, j))$; j++ ▷ ASR beam search
  while $\phi_\pi(B) - k \geqslant t$ **do** ▷ new tokens?
    $\hat{y}_{t+1} \leftarrow p^{\text{ST}}_{\text{wait-}k}(y_{t+1} \mid \bar{\mathbf{s}}_{[1:i+1]}, \hat{\mathbf{y}}_{\leq t}; \boldsymbol{\theta}^{\text{ST}}_{\text{full}})$
    yield $\hat{y}_{t+1}$; t++ ▷ commit translation to user

- **Longest Common Prefix** (LCP) uses the length of longest shared prefix in the streaming ASR beam as the number of valid tokens within given speech. This is the most conservative strategy, which has similar latency to cascaded methods.
- **Shortest Hypothesis** (SH) uses the length of shortest hypothesis in the current streaming ASR beam as the number of valid tokens.

More formally, let $\phi_\pi(B)$ denote the number of valid tokens in the beam B under policy π:

$$
\begin{aligned}
\phi_{\text{LCP}}(B) &= \max\{i \mid \exists \mathbf{z}', s.t. \forall \langle \mathbf{z}, c \rangle \in B, \mathbf{z}_{\leq i} = \mathbf{z}'\} \\
\phi_{\text{SH}}(B) &= \min\{|\mathbf{z}| \mid \langle \mathbf{z}, c \rangle \in B\}
\end{aligned}
$$

For example in Fig. 3, $\phi_{\text{LCP}}(B_7) = 3$, $\phi_{\text{SH}}(B_7) = 5$. Also note that $\phi_{\text{LCP}}(B) \leq \phi_{\text{SH}}(B)$ for any beam B, and that both policies are *monotonic*, i.e. $\phi_\pi(B_j) \leq \phi_\pi(B_{j+1})$ for $\pi \in \{\text{LCP}, \text{SH}\}$ and all j.

Note we always feed the entire observable speech segments into ST for translation, and streaming ASR-generated transcription is not used for translation, so LCP might have similar latency with cascaded methods but the translation accuracy is much better because more information on the source side is revealed to the translation decoder.

As shown in Algorithm 1, during simultaneous ST, we monitor the value of $\phi_\pi(B_j)$ while speech

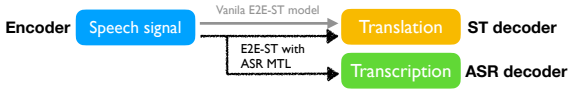

Figure 4: We use full-sentence MTL framework to jointly learn ASR and ST with a shared encoder.

chunks are gradually fed into system. When we have $\phi_\pi(B) - k \geqslant t$ where t is the number of translated tokens, the ST decoder will be triggered to generate one new token as follows:

$$\hat{y}_t = \underset{y_t}{\mathbf{argmax}}\, p_{\text{wait-}k}(y_t \mid \bar{\mathbf{s}}_{[1:\tau(j)]}, \hat{\mathbf{y}}_{<t}; \hat{\boldsymbol{\theta}}^{\text{ST}}_{\text{full}}) \quad (2)$$

3.2 Joint Training between ST and ASR

Different from existing simultaneous translation solutions from (Ren et al., 2020; Ma et al., 2020b,a), which make adaptations over vanilla E2E-ST architecture as shown in gray line of Fig. 4, we instead use simple MTL architecture which performs joint full-sentence training between ST and ASR:

$$\hat{\boldsymbol{\theta}}^{\text{ST}}_{\text{full}}, \hat{\boldsymbol{\theta}}^{\text{ASR}}_{\text{full}} = \underset{\boldsymbol{\theta}^{\text{ST}}_{\text{full}}, \boldsymbol{\theta}^{\text{ASR}}_{\text{full}}}{\mathbf{argmax}} \prod_{(\mathbf{s}, \mathbf{y}^*, \mathbf{z}^*) \in D} p^{\text{ST}}_{\text{full}}(\mathbf{y}^* \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ST}}_{\text{full}}) \cdot p^{\text{ASR}}_{\text{full}}(\mathbf{z}^* \mid \mathbf{s}; \boldsymbol{\theta}^{\text{ASR}}_{\text{full}})$$

For ASR training, we use hybrid CTC/Attention framework (Watanabe et al., 2017). Note that we train ASR and ST MTL with full-sentence fashion for simplicity and training efficiency, and only perform wait-k decoding policy at inference time. Also, $\boldsymbol{\theta}^{\text{ST}}_{\text{full}}$ and $\boldsymbol{\theta}^{\text{ASR}}_{\text{full}}$ share the same speech encoder.

4 Experiments

We conduct experiments on English-to-German (En→De) and English-Spanish (En→Es) translation on MuST-C (Di Gangi et al., 2019). We employ Transformer (Vaswani et al., 2017) as the basic architecture and LSTM (Hochreiter and Schmidhuber, 1997) for LM. For streaming ASR decoding we use a beam size of 5. Translation decoding is greedy due to incremental commitment.

Raw audios are processed with Kaldi (Povey et al., 2011) to extract 80-dimensional log-Mel filterbanks stacked with 3-dimensional pitch features using a 10ms step size and a 25ms window size. Text is processed by SentencePiece (Kudo and Richardson, 2018) with a joint vocabulary size of 8K. We take Transformer (Vaswani et al., 2017) as our base architecture, which follows 2 layers of 2D convolution of size 3 with stride size of 2. The Transformer model has 12 encoder layers and

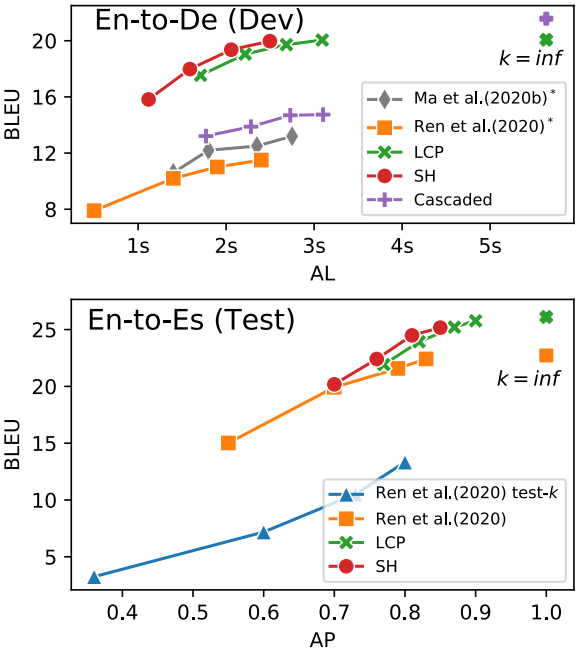

Figure 5: Translation quality v.s. latency. The dots on each curve represents different wait-k policy with k=1,3,5,7 from left to right respectively. Baseline* results are from Ma et al. (2020b). k=inf is full-sentence decoding for ASR and translation. test-k denotes testing time wait-k. We use a chunk size of 48.

6 decoder layers. Each layer has 4 attention head with a size of 256. Our streaming ASR decoding method follows Moritz et al. (2020). We employ 10 frames look ahead for all experiments. For LM, we use 2 layers stacked LSTM (Hochreiter and Schmidhuber, 1997) with 1024-dimensional hidden states, and set the embedding size as 1024. LM are trained on English transcription from the corresponding language pair in MuST-C corpus. For the cascaded model, we train ASR and MT models on Must-C dataset respectively, and they have the same Transformer architecture of our ST model. Our experiments are run on 8 1080Ti GPUs. And the we report the case-sensitive detokenized BLEU.

Translation quality against latency In order to clearly compare with related works, we evaluate the latency with AL defined in Ma et al. (2020b) and AP defined in Ren et al. (2020). As shown in Fig. 5, for En→De, results are on the dev set to be consistent with Ma et al. (2020b). Compared with baseline models, our method achieves much better translation quality with similar latency. To validate the effectiveness of our method, we compare our method with Ren et al. (2020) on En→Es translation. Their method does not evaluate the

chunk index	1	2	3	4	5	6	end
Gold transcript	can I be	**honest**	*SIL*	I don 't **love**	**that question**	*SIL*	
Gold translation	Darf ich **ehrlich** sein ?			Ich **mag diese Frage** nicht .			
Streaming ASR	can I			be **on this** I don 't	**love that question**		
simul-MT wait-3				Kann ich **da** sein ? "	Ich **liebe**		**diese Frage** nicht .
SH wait-3				Kann ich **ehrlich** sein ? Ich **liebe**	**diese Frage**		nicht .
LCP wait-3				Kann ich **ehrlich** sein ? Ich			**liebe diese Frage** nicht .

Figure 6: An example from the dev set of En→De translation. In the cascaded approach (streaming ASR + simul-MT wait-3), the ASR error (*"on this"* for *"honest"*) is propagated to the MT module, causing the wrong translation (*"da"*). Our methods give accurate translations ("ehrlich") with better latency (esp. for the SH policy, the output of *"diese Frage"* is synchronous with hearing *"that question"*). *"SIL"* denotes silence in speech.

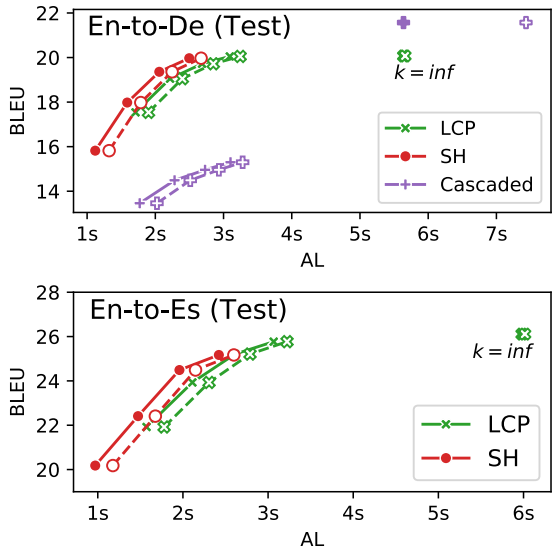

Figure 7: Translation quality against latency. Each curve represents decoding with wait-k policy, k=1,3,5,7 from left to right. The dashed lines and hollow markers indicate the latency considering the computational time. The chunk size is 48.

plausibility of the detected tokens, so it has a more aggressive decoding policy which results in lower latency. However, our method can still achieve better results with slightly lower latency. Besides that, our model is trained in full-sentence mode, and only decodes with wait-k at inference time, which is very efficient to train. Our test-time wait-k could achieve similar quality with their genuine wait-k (i.e., retrained) models which are very slow to train. When we compare with their test-time wait-k, our model significantly outperforms theirs.

We further evaluate our method on the test set of En→De and En→Es translation. As shown in Fig. 7, compared with the cascaded model, our model has notable successes in latency and translation quality. To verify the online usability of our model, we also show computational-aware latency. Because our chunk window is 480ms, and the latency caused by the computation is smaller than this window size, which means that we can finish decoding the previous speech chunk when the next speech chunk needs to be processed, so our model can be effectively used online.

Model	En→De			En→Es		
	$w=32$	$w=48$	$w=64$	$w=32$	$w=48$	$w=64$
LCP	17.31	17.54	17.95	21.94	21.92	22.36
− *LM*	14.60	15.66	15.91	18.54	19.15	19.95
− *LM & AD*	13.76	14.82	15.26	17.42	18.06	19.32
SH	16.04	15.82	15.87	20.45	20.18	19.84
− *LM*	13.76	14.01	13.84	17.31	17.21	17.78
− *LM & AD*	10.44	11.25	11.65	13.61	14.27	14.62

Table 1: BLEU score of wait-1 decoding with different chunk sizes and ASR scoring functions. *AD* denotes ASR Decoder. *LM* denotes Language Model.

Fig. 6 demonstrates that our method can effectively avoid the error propagation and obtain better latency compared to the cascaded model.

Effect of chunk size and joint decision Table 1 shows that the results are relatively stable with various chunk sizes. It can be flexible to balance the response frequency and computational ability. We explore the effectiveness of ASR joint scoring, and observe that the translation quality drops a lot without LM. Without LM and AD, our token recognition approach is similar to the speech segmentation in Ren et al. (2020), which implies that their model is hard to segment the source speech accurately, leading to unreliable translation decisions for ST.

5 Conclusion

We proposed a simple but effective ASR-assisted simultaneous E2E-ST framework. The streaming ASR module can guide (but not give direct input to) the wait-k policy for simultaneous translation. Our method improves ST accuracy with similar latency.

Acknowledgments

This work is supported in part by NSF IIS-1817231 and IIS-2009071.

References

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. *Meeting of the Association for Computational Linguistics*.

Naveen Arivazhagan, Colin Cherry, Isabelle Te, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2020. Re-translation strategies for long form, simultaneous, spoken language translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7919–7923. IEEE.

Junkun Chen, Mingbo Ma, Renjie Zheng, and Liang Huang. 2020. Mam: Masked acoustic modeling for end-to-end speech-to-text translation. *arXiv preprint arXiv:2010.11445*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium. Association for Computational Linguistics.

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. 2019. 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*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020a. SimulMT to SimulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*.

Xutai Ma, Yongqiang Wang, Mohammad Javad Dousti, P. Koehn, and J. Pino. 2020b. Streaming simultaneous speech translation with augmented memory transformer. *ArXiv*, abs/2011.00033.

Niko Moritz, Takaaki Hori, and Jonathan Le. 2020. Streaming automatic speech recognition with the transformer model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6074–6078. IEEE.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Nagendra Goel, Mirko Hannemann, Yanmin Qian, Petr Schwarz, and Georg Stemmer. 2011. The kaldi speech recognition toolkit. In *In IEEE 2011 workshop*.

Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. SimulSpeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Chengyi Wang, Yu Wu, Shujie Liu, Jinyu Li, Liang Lu, Guoli Ye, and Ming Zhou. 2020. Low latency end-to-end streaming speech recognition with a scout network. *arXiv preprint arXiv:2003.10369*.

S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi. 2017. Hybrid ctc/attention architecture for end-to-end speech recognition.

Hao Xiong, Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Dutongchuan: Context-aware translation model for simultaneous interpreting. *arXiv preprint arXiv:1907.12984*.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020a. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. Simultaneous translation with flexible policy via restricted imitation learning. In *ACL*.

Renjie Zheng, Junkun Chen, Mingbo Ma, and Liang Huang. 2021. Fused acoustic and text encoding for multimodal bilingual pretraining and speech translation. *Proceedings of the 38th International Conference on Machine Learning*.

Renjie Zheng, Mingbo Ma, Baigong Zheng, Kaibo Liu, Jiahong Yuan, Kenneth Church, and Liang Huang. 2020b. Fluent and low-latency simultaneous speech-to-speech translation with self-adaptive training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3928–3937.