Incremental Blockwise Beam Search for Simultaneous Speech Translation with Controllable Quality-Latency Tradeoff

Peter Polák[1], Brian Yan[2], Shinji Watanabe[2], Alex Waibel[2], Ondřej Bojar[1]

[1]Charles University, Czechia
[2]Carnegie Mellon University, USA

polak@ufal.mff.cuni.cz

Abstract

Blockwise self-attentional encoder models have recently emerged as one promising end-to-end approach to simultaneous speech translation. These models employ a blockwise beam search with hypothesis reliability scoring to determine when to wait for more input speech before translating further. However, this method maintains multiple hypotheses until the entire speech input is consumed – this scheme cannot directly show a single *incremental* translation to users. Further, this method lacks mechanisms for *controlling* the quality vs. latency tradeoff. We propose a modified incremental blockwise beam search incorporating local agreement or hold-n policies for quality-latency control. We apply our framework to models trained for online or offline translation and demonstrate that both types can be effectively used in online mode.

Experimental results on MuST-C show 0.6-3.6 BLEU improvement without changing latency or 0.8-1.4 s latency improvement without changing quality.

Index Terms: simultaneous speech translation, beam search decoding, blockwise encoder

1. Introduction

Simultaneous speech translation (SST) is the task of translating speech into a different language before the utterance is finished. Traditionally, this task has been addressed by a cascade of automatic speech recognition (ASR) and machine translation (MT) [1]. More recently, E2E approaches have emerged, demonstrating reduced latency [2, 3].

The ultimate goal of SST is a real-time user experience, i.e., the goal is not only maximal translation quality but also minimal latency. Several solutions for this problem have been proposed, for example, the wait-k [4] policy which limits the number of emitted tokens by the number of valid source tokens. However, wait-k cannot directly use beam search, and its direct application to speech input is also complicated [5]. Alternatively, we can leave the model to generate the whole translation for the current context and heuristically decide which portion of the translation is reliable [6, 7]. However, relying on attention can lead to over-generation and low-quality translation [7, 8].

Other approaches include more flexible solutions that leave the decision of how much input to read before generating translations to the model. One such approach is monotonic (chunkwise) attention [9, 10]; however, these methods rely on strong monotonic restrictions that may limit the performance. Another such approach is blockwise self-attentional encoder models with blockwise beam search (BWBS) [11, 12]. This approach is advantageous in that blockwise processing reduces computational complexity for encoder networks. Further, BWBS performs an adaptive inference in which a hypothesis reliability score is used to determine whether the decoding should wait for more input speech in order to produce a higher quality translation [11]. However, BWBS, which was originally proposed for ASR, lacks several key characteristics which are commonly desired in SST applications.

In particular, we are interested in BWBS models that 1) produce *incremental* translations (see Section 2.2 and Figure 1) and 2) have mechanisms for *controlling* the quality vs. latency tradeoff during inference. The previously proposed BWBS scheme maintains multiple hypotheses until the entire speech input is consumed, while SST systems are often expected to show only one translation result, which is gradually incremented to the user. The previously proposed BWBS scheme also only controls the quality-latency by changing the block duration, which may require training a new model and is not a fine-grained control mechanism.

We propose an incremental blockwise beam search (IBWBS) using local agreement or hold-n policies for quality-latency control. Our IBWBS algorithm also modifies the stopping criterion of the original BWBS, which we found to be overly conservative for translation. Instead of stopping the whole beam search when an unreliable hypothesis is detected, we stop only the affected beam, and we continue to expand the remaining beams. We apply IBWBS to models with limited (e.g. contextual block [11]) and full-context encoders. The original BWBS used only contextual block encoders, but we extend to full-context encoders as well, demonstrating that this framework can onlinize models with conventionally offline architectures. Our experiments on the MuST-C corpus show an improvement of 0.6-3.6 BLEU without changing latency for contextual block models, and latency improvement of 0.8-1.4 sec for full-context models with IBWBS compared to the original BWBS. Additionally, we show that the proposed IBWBS improves the translation quality by 5-8 BLEU when used with the local agreement policy or lowers the computational complexity by 20-30 % when used with the hold-n policy.

2. Background

In this section, we first review blockwise processing for SST. We then describe the differences between re-translation and incremental models. Finally, we review quality-latency controls.

2.1. Blockwise Streaming Encoder and Beam Search

Previous studies have shown that block processing can be an effective way to reduce the computational complexity for on-
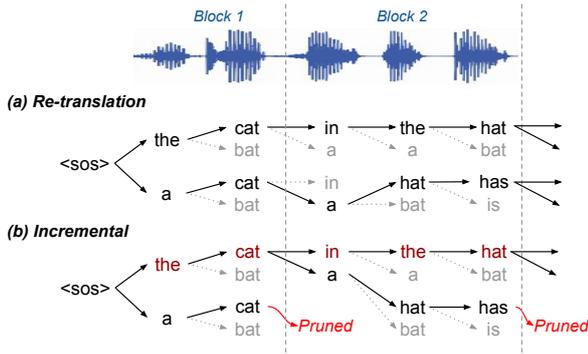
Figure 1: *Re-translation vs. Incremental Decoding.*

line Encoder [11–14] for speech recognition and translation. Specifically, the source speech is split into blocks of equal size [11]. Each block is encoded using the block's acoustic features and a contextual embedding inherited from the previous block. The encoded source features of the i-th block are $B^i = (B^i_1, ..., B^i_T)$, where T is block size.

To achieve a simultaneous translation, [11] proposes a blockwise streaming beam search (BWBS; see Algorithm 1). Unlike traditional beam search [15, 16], the **blockwise streaming beam search predicts the** y_j **based on** the previous predictions $y_{1:j-1}$ and **unfinished source features** $B^{1:b}$. To ensure that the model does not hallucinate translation beyond current source context $B^{1:b}$, [11] detects a repeated tokens or end-of-sequence token (`<eos>`) in any of the hypotheses (see Line 5). They base this heuristic on an observation that attentional Encoder-Decoder models tend to repeat tokens [8] or prematurely end when presented with insufficient source context. Once a repetition or `<eos>` token is detected, the last two tokens are removed (Line 6) and the algorithm waits for more source (Line 7).

Algorithm 1: Blockwise streaming beam search [11]

    Input : A list of blocks, blockwise ST model
    Output: A set of hypotheses and scores
1 **for** *each block* **do**
2   Encode block using the blockwise ST model;
3   **while** *not maximum length* **do**
4   Extend beams and compute scores;
5   **if** *any beam contains a repetition or* `<eos>` **then**
6   Remove last two tokens from every beam;
7   Break;
8   **end**
9   **end**
10 **end**

2.2. Incremental vs. Re-translation Models

SST models can be classified as either re-translation or incremental models (as shown in Figure 1). Re-translation models [17, 18] maintain multiple hypotheses throughout the entire decoding. From a user perspective, these systems would display either the top hypothesis or a set of hypotheses – critically, a re-translation model is capable of revising the top hypothesis or re-ranking the set of hypotheses as more speech input is read. This design arguably makes it more difficult for the user to process the translation and it also makes it more difficult to evaluate the latency of the model.

In fact, many SST latency metrics [19] were originally designed for only incremental translation.[1] Incremental models [21, 22] differ from re-translation models in that they prune all hypotheses to a common prefix which is then shown to the user. For the user, the translation changes only by incrementally getting longer; none of the previously displayed outputs are ever modified. In this work, we focus on incremental SST models.

To this end, we note that **the blockwise streaming beam search described in Section 2.1 is, in fact, a re-translation SST decoding algorithm**. Specifically, when the decoding is stopped due to insufficient context (see Line 5 in Algorithm 1), the beam search contains different beams, i.e., they do not share the same prefix. In the future steps, any of these beams can be either extended or removed from the search (by expanding into a lower-scoring hypothesis that falls out of the beam search), which prevents us from presenting a single stable hypothesis to the user.

2.3. Latency-Quality Trade-off

Simultaneous speech translation aims to produce a high-quality translation in real-time with low latency. However, these two objectives are conflicting. If we decrease the latency, the translation quality also drops. To control this latency-quality trade-off in incremental SST, we use a policy that decides how much translation to produce for the current input [4, 6, 21–26].

For blockwise models (see Section 2.1), the sole means of controlling latency has been varying block size. Although this method is simple, its primary drawback is the need to train a new model for each latency regime. Additionally, reducing the block size may not be ideal for translation as the context quickly becomes too little.

To onlinize **full-context models**, [6] proposed chunking,[2] i.e., splitting the source speech into segments that are incrementally fed into the model. **Latency is controlled by selecting only a part of the model's output**. Specifically, [6] proposed a hold-n policy that removes the last n tokens from the model output, and local agreement that finds the longest common prefix of outputs obtained for two consecutive input contexts. Moreover, [7] showed that varying the chunk size can also be effectively applied along these policies.

suTo generate the translation for the partial input, the full-context models leverage a standard beam search known from machine translation [15]. The advantage of the standard beam search is that the model can generate a complete translation for current speech input. However, it is also prone to overgeneration and low-quality translations toward the end of the context [7]. In ASR, the **standard beam search with attentional models shown poor length generalization** [27].

3. Proposed Method

In this section, we first propose an incremental variant of the BWBS algorithm introduced in Section 2.1. We further augment our approach with quality-latency controls and propose an alternative hypothesis expansion strategy, which avoids overly conservative decoding. Finally, we describe how our framework applies to not only limited-context but also full-context models.

---

[1]IWSLT shared tasks [2, 3, 20] also follow this evaluation standard.
[2]We use "chunk" for full-context models and "block" for blockwise models.

3.1. Controllable Latency for Incremental SST with Block-wise Models

Incorporating incremental pruning into our algorithm involves performing the pruning step after processing each block (see Line 15 in Algorithm 2). We select the highest-scoring hypothesis and discard the others, resulting in only one translation prefix remaining after each block has been processed.

Second, we apply an incremental policy to facilitate latency control, allowing us to use a single trained model in multiple latency regimes without having to retrain it for different block sizes. Specifically, we apply the hold-n or local agreement policies as discussed in Section 2.3.

3.2. Improved Blockwise Streaming Beam Search

As described in Section 2.1, the blockwise streaming beam search enables translation of unfinished utterances while ensuring that the model stays within the current source context. However, if any beam contains a repetition or the end-of-sequence token, the entire beam search is halted (Line 7 in Algorithm 1).

To avoid this overly conservative behavior, we relax the stopping criterion. We outline the algorithm in Algorithm 2. Instead of stopping the whole beam search, we only stop beams with a repetition or `<eos>` (Line 10). This ensures that each individual beam is expanded until it hits the stopping criterion. Once all the beams are *stopped*, we stop expanding the hypothesis for the current block. Following the incremental pruning described in Section 3.1, we select the best beam from the stopped beams (Line 15). It is important to note that the stopped beams may contain hypotheses of different lengths. Therefore, when comparing the finished hypotheses based on score (Line 14), we apply length normalization [28, 29].

Algorithm 2: Proposed improved blockwise streaming beam search algorithm for incremental ST

Input : A list of blocks, blockwise ST model
Output: A set of hypotheses and scores

1 **for** *each block* **do**
2   Encode block using the blockwise ST model;
3   *Stopped* ← ∅;
4   **while** *#active beams > 0 and not max. length* **do**
5   Extend beams and compute scores;
6   **for** *each active beam b* **do**
7   **if** *b contains a repetition or* `<eos>` **then**
8   Remove the last two tokens from b;
9   *Stopped* ← *Stopped* ∪ b;
10   Remove b from the beam search;
11   **end**
12   **end**
13   **end**
14   Sort *Stopped* by length-normalized score;
15   Set the best hypothesis from *Stopped* as active beam;
16   Apply incremental policy;   // Hold-n or LA
17 **end**

3.3. Blockwise Streaming Beam Search for Full-Context Models

Finally, instead of standard beam search, we apply the proposed blockwise streaming beam search to the full-context offline models. To do so, we implement the onlinization framework by [6] described in Section 2.3. Since we use offline models, we recompute the encoder and decoder states after each new chunk.

4. Experiments

4.1. Data

In our experiments, we use the English → German, a Subject-Verb-Object (SVO) language to SOV language pair, English → Spanish, an SVO-SVO language pair, and English → French, an SVO-SVO language pair, of the MuST-C [30] data set. We use the training and validation sets during the training of the blockwise models. Finally, we use the `tst-COMMON` split for the evaluation of the online decoding algorithms.

4.2. Models

For the blockwise speech translation models, we use the ESP-Net toolkit [31]. We preprocess the audio with 80-dimensional filter banks. For both language pairs, we built a unigram [32] vocabulary with a size of 4000. We build three models with block sizes of 20, 32, and 40 for each language pair. The encoder has 12 layers, and the decoder has 6 layers. The model dimension is 256, feed-forward dimension is 2048 with 4 attention heads. To improve the training speed, we initialize the encoder with weights pretrained on the ASR task of the MuST-C dataset. Further, we employ ST CTC [12, 33] after the encoder with weight 0.3. However, we do not use the CTC loss during inference. Additionally, we employ checkpoint averaging for the last 10 epochs.

For the offline ST models, we use encoder-decoder architecture based on Transformer. Specifically, we use the pretrained offline models introduced in [34].[3] The models are implemented in Fairseq [35]. The encoder is based on wav2vec 2.0 [36]; therefore, the models' input is raw single-channel speech with 16k sampling frequency.

All models are evaluated using Simuleval [19] toolkit. For the translation quality, we report detokenized case-sensitive BLEU [37], and for the latency, we report length-aware average lagging (LAAL) [7, 38]. In all our experiments, we use beam search with size 6. For the hold-n strategy with the offline models, we use a fixed step size of 280 ms [5]. Additionally, we remove the repetition detection for the offline models. Our initial experiments showed that the offline models do not generate repetitions. On the other hand, the blockwise models are prone to generate repetitions; therefore, we keep the repetition detection on for all blockwise experiments.

Lang	Re-translation	Incremental	Δ
En-De	23.0	22.4	-0.6
En-Es	27.9	27.2	-0.7
En-Fr	31.5	30.8	-0.7

Table 1: *Comparison of the re-translation and incremental translation approach in terms of BLEU on* `tst-COMMON`.

4.3. Incremental Blockwise Beam Search

In our first attempt on incremental BWBS, we apply incremental pruning after each processed block, but without the incremental policies (see Section 3.1). In Table 1, we compare the performance between re-translation and incremental SST. The difference is between 0.6 and 0.7 BLEU in favor of re-translation SST – we deem this to be an acceptable degradation in order to enable the incremental user experience. Note that since SimulEval latency evaluation expects an incremental output, we do not evaluate the latency of re-translation models.

(a) *Latency control using block size.*   (b) *Latency control using incremental policies.*   (c) *Proposed improved IBWBS.*
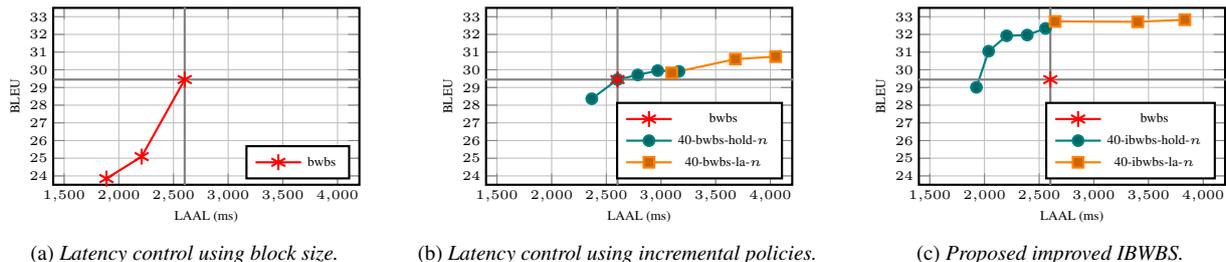
Figure 2: *English → French blockwise model.*

Lang	Decoding	LAAL	BLEU
	Blockwise Models		
En-De	BWBS	2433	23.2
	IBWBS	2355	**23.8**
En-Es	BWBS	2303	26.1
	IBWBS	2335	**27.0**
En-Fr	BWBS	2367	28.4
	IBWBS	2390	**32.0**
	Full-Context Models		
En-De	BWBS	2838	28.0
	IBWBS	**1879**	27.6
En-Es	BWBS	4073	33.1
	IBWBS	**2678**	33.0
En-Fr	BWBS	3420	38.8
	IBWBS	**2668**	38.6

Table 2: *Incremental SST with the original BWBS and the proposed IBWBS.*

Lang	Decoding	# fw. passes	LAAL	BLEU
En-De	BS	729,091	1676	21.4
	IBWBS	**583,787**	1879	**27.6**
En-Es	BS	749,272	1643	25.7
	IBWBS	**608,664**	1839	**31.4**
En-Fr	BS	837,084	1665	29.0
	IBWBS	**675,683**	1854	**37.3**

Table 3: *Comparison of the standard beam search [15] (BS) and the proposed IBWBS with local agreement policy using onlinized full-context models.*

4.4. Controllable Latency for Blockwise Encoder

In Figure 2a, we show the latency control using the block size. Each point on the BWBS line corresponds to a single model. As described in Section 3.1, we apply incremental policies to facilitate latency control. In Figure 2b, we apply the incremental pruning with hold-n and local agreement policies to the model with block 40, which allows for a wide range of latencies.

4.5. Improved Blockwise Streaming Beam Search

In Figure 2c, we illustrate the benefits of the proposed improved BWBS (see Section 3.2) on En-Fr. Compared to the original BWBS in Figure 2b, we observe improvements in both quality (by more than 2 BLEU) and latency (by 500 ms).

Results on all languages are in Table 2. For the blockwise model, we select systems with a latency of approx. 2300 ms and observe a quality improvement of 0.6, 0.9, and 3.6 BLEU for

Lang	Decoding	# fw. passes	LAAL	BLEU
En-De	BS	1,210,691	1693	25.5
	IBWBS	**946,088**	1704	25.4
En-Es	BS	1,214,720	1696	29.7
	IBWBS	**993,124**	1653	29.0
En-Fr	BS	1,358,738	1602	33.8
	IBWBS	**1,022,294**	1604	33.7

Table 4: *Comparison of the standard beam search [15] (BS) and the proposed IBWBS with hold-n policy using onlinized full-context models.*

English to German, Spanish, and French, respectively. For the onlinized models, we select models with similar BLEU scores because original BWBS decoding has much higher latencies compared to the proposed IBWBS. In Table 2, we observe latency improvements of 959, 1395, and 752 milliseconds for English to German, Spanish, and French, respectively.

4.6. Improved Blockwise Streaming Beam Search for Full-Context Models

In this section, we compare the standard beam search with the proposed IBWBS (see Section 3.3). For the local agreement policy, we present the results in Table 3. For systems with a latency of approx. 1.7 sec, we observe a quality improvement of 6.2, 5.7, and 8.3 BLEU for En-De, En-Es, and En-Fr, respectively. Additionally, we measure the computational complexity. To avoid hardware-specific evaluation, we report the number of forward passes through the decoder. From the middle column in Table 3, we see the IBWBS helps reduce the computational complexity by approx. 20 %.

In Table 4, we compare the proposed IBWBS and the standard beam search for the hold-n policy. Here, we do not observe any quality or latency change. However, similarly to the local agreement, we observe a computational complexity reduction of 20 to 30 % across all languages.

5. Conclusion

In this paper, we take a deep dive into blockwise encoders and blockwise streaming beam search for simultaneous speech translation. We propose a modified incremental blockwise beam search, which incorporates local agreement or hold-n policies for quality-latency control. This enables incremental SST and facilitates latency control without the need to retrain the model. Furthermore, we document that the proposed changes bring improvement to both quality and latency. Additionally, we show our framework can be directly applied to full-context encoders, which leads to quality and performance improvements compared to the standard beam search.

6. References

[1] C. Fügen, A. Waibel, and M. Kolss, "Simultaneous translation of lectures and speeches," *Machine translation*, vol. 21, pp. 209–252, 2007.

[2] A. Anastasopoulos *et al.*, "FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN," in *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, 2021, pp. 1–29.

[3] A. Anastasopoulos *et al.*, "FINDINGS OF THE IWSLT 2022 EVALUATION CAMPAIGN," in *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, 2022.

[4] M. Ma *et al.*, "Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework," in *Proc. ACL*, 2019, pp. 3025–3036.

[5] X. Ma, J. Pino, and P. Koehn, "SimulMT to SimulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation," in *Proc. ACL*, 2020, pp. 582–587.

[6] D. Liu, G. Spanakis, and J. Niehues, "Low-Latency Sequence-to-Sequence Speech Recognition and Translation by Partial Hypothesis Selection," in *Proc. Interspeech*, 2020, pp. 3620–3624.

[7] P. Polák *et al.*, "CUNI-KIT System for Smultaneous Speech Translation Task at IWSLT 2022," in *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, 2022.

[8] S. Watanabe *et al.*, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[9] X. Ma *et al.*, "Monotonic multihead attention," *arXiv preprint arXiv:1909.12406*, 2019.

[10] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *Proc. ICLR*, 2017.

[11] E. Tsunoo, Y. Kashiwagi, and S. Watanabe, "Streaming transformer asr with blockwise synchronous beam search," in *Proc. SLT*, 2021, pp. 22–29.

[12] K. Deng *et al.*, "Blockwise streaming transformer for spoken language understanding and simultaneous speech translation," *arXiv preprint arXiv:2204.08920*, 2022.

[13] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *Proc. ICASSP*, 2020, pp. 6074–6078.

[14] Y. Shi *et al.*, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *Proc. ICASSP*, 2021, pp. 6783–6787.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[17] J. Niehues *et al.*, "Dynamic transcription for low-latency speech translation," in *17th Annual Conference of the International Speech Communication Association, INTERSPEECH 2016; Hyatt Regency San FranciscoSan Francisco; United States; 8 September 2016 through 16 September 2016*, ser. Proceedings of the Annual Conference of the International Speech Communication Association. Ed. : N. Morgan, vol. 08-12-September-2016, 2016, pp. 2513–2517.

[18] J. Niehues *et al.*, "Low-latency neural speech translation," in *19th Annual Conference of the International Speech Communication, INTERSPEECH 2018; Hyderabad International Convention Centre (HICC)Hyderabad; India; 2 September 2018 through 6 September 2018. Ed.: C.C. Sekhar*, ser. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, vol. 2018-September, 2018, pp. 1293–1297.

[19] X. Ma *et al.*, "Simuleval: An evaluation toolkit for simultaneous translation," in *Proc. EMNLP*, 2020, pp. 144–150.

[20] E. Ansari *et al.*, "FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN," in *Proceedings of the 17th International Conference on Spoken Language Translation*, 2020, pp. 1–34.

[21] K. Cho and M. Esipova, "Can neural machine translation do simultaneous translation?" *arXiv preprint arXiv:1606.02012*, 2016.

[22] F. Dalvi *et al.*, "Incremental decoding and training methods for simultaneous translation in neural machine translation," in *Proc. ACL*, 2018, pp. 493–499.

[23] J. Gu *et al.*, "Learning to translate in real-time with neural machine translation," in *Proc. ACL*, 2017, pp. 1053–1062.

[24] N. Arivazhagan *et al.*, "Monotonic infinite lookback attention for simultaneous machine translation," in *Proc. ACL*, 2019, pp. 1313–1323.

[25] B. Zheng *et al.*, "Simpler and faster learning of adaptive policies for simultaneous translation," in *Proc. EMNLP*, 2019, pp. 1349–1354.

[26] B. Zheng *et al.*, "Simultaneous translation with flexible policy via restricted imitation learning," in *Proc. ACL*, 2019, pp. 5816–5822.

[27] L. Dong *et al.*, "A comparison of label-synchronous and frame-synchronous end-to-end models for speech recognition," *arXiv preprint arXiv:2005.10113*, 2020.

[28] S. Jean *et al.*, "Montreal neural machine translation systems for wmt'15," in *Proceedings of the tenth workshop on statistical machine translation*, 2015, pp. 134–140.

[29] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks.," in *ISMIR*, 2013, pp. 335–340.

[30] R. Cattoni *et al.*, "Must-c: A multilingual corpus for end-to-end speech translation," *Computer Speech & Language*, vol. 66, p. 101 155, 2021.

[31] H. Inaguma *et al.*, "ESPnet-ST: All-in-one speech translation toolkit," in *Proc. ACL*, 2020, pp. 302–311.

[32] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proc. ACL*, 2018, pp. 66–75.

[33] B. Yan *et al.*, "Ctc alignments improve autoregressive translation," *arXiv preprint arXiv:2210.05200*, 2022.

[34] Y. Tang *et al.*, "Unified speech-text pre-training for speech translation and recognition," in *Proc. ACL*, 2022, pp. 1488–1499.

[35] M. Ott *et al.*, "Fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[36] A. Baevski *et al.*, "Wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[37] M. Post, "A call for clarity in reporting bleu scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018, pp. 186–191.

[38] S. Papi *et al.*, "Over-generation cannot be rewarded: Length-adaptive average lagging for simultaneous speech translation," in *Proceedings of the Third Workshop on Automatic Simultaneous Translation*, 2022, pp. 12–17.