Interspeech 2024
1-5 September 2024, Kos, Greece

# Simul-Whisper: Attention-Guided Streaming Whisper with Truncation Detection

*Haoyu Wang*[1†], *Guoqiang Hu*[2†], *Guodong Lin*[1†], *Wei-Qiang Zhang*[1*], *Jian Li*[3]

[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2]International School, Jinan University, Guangzhou, 511443, China
[3]Beijing Sinovoice Technology Co.,Ltd, China

w-hy21@mails.tsinghua.edu.cn, wq-zhang@tsinghua.edu.cn

## Abstract

As a robust and large-scale multilingual speech recognition model, Whisper has demonstrated impressive results in many low-resource and out-of-distribution scenarios. However, its encoder-decoder structure hinders its application to streaming speech recognition. In this paper, we introduce **Simul-Whisper**, which uses the time alignment embedded in Whisper's cross-attention to guide auto-regressive decoding and achieve chunk-based streaming ASR without any fine-tuning of the pre-trained model. Furthermore, we observe the negative effect of the truncated words at the chunk boundaries on the decoding results and propose an integrate-and-fire-based truncation detection model to address this issue. Experiments on multiple languages and Whisper architectures show that **Simul-Whisper** achieves an average absolute word error rate degradation of only 1.46% at a chunk size of 1 second, which significantly outperforms the current state-of-the-art baseline.

**Index Terms**: streaming speech recognition, Whisper, decision policy, truncation detection

## 1. Introduction

Pre-training on large-scale datasets has brought significant improvements in automatic speech recognition (ASR) [1, 2]. Recently, Whisper [3], a weakly supervised encoder-decoder model trained on a large dataset of 680,000 hours, was shown to have remarkably robust performance across different languages and complex open environments [4, 5, 6, 7]. On the other hand, Whisper is not pre-trained to perform streaming ASR, which sets a barrier to its application in scenarios such as conference transcription, simultaneous translation, and live streaming. This makes using Whisper for streaming ASR an attractive proposition.

Compared to offline ASR, streaming ASR is more challenging due to the lack of full context during inference. Transformer-based streaming ASR models typically use time-restricted [8], chunk-wise [9], or memory-based [10] attention. However, modifying the pre-trained parameters to use such streaming attention mechanisms requires large computational resources [11]. Besides, compared to encoder-only pre-trained models such as HuBERT [12] or WavLM [13], Whisper models face additional challenges when used for streaming ASR due to their encoder-decoder structures. In an encoder-decoder structure, the encoder converts the input audio into latent representations all at once, after which the decoder begins iterative auto-regressive decoding until it encounters a special " ⟨eos⟩" token in the output sequence. In streaming ASR, the input to the model is often truncated into short segments of fixed duration. This random truncation can lead to unreliable outputs at the end of the transcription, and can even cause attention to fail, resulting in meaningless repetitive outputs. These errors cannot be avoided by stopping decoding before the boundary, since the start and end timestamps corresponding to the output tokens cannot be found due to the non-monotonic nature of the encoder-decoder cross-attention.

To address these issues, Macháček et al. propose to apply the Local Agreement [14] policy to the Whisper models. They use a chunk-based inference approach, where Whisper performs non-streaming decoding each time a chunk of audio is received, and the final transcription is determined by the longest common prefix of the results from previous chunks. This method avoids modifying the parameters of Whisper, while mitigating unreliable transcriptions at chunk boundaries. However, since the model only generates transcriptions when the longest common prefix grows, the latency is usually unpredictable. Additionally, repeatedly performing non-streaming decoding is also computationally intensive.

Streaming inference of encoder-decoder models is also an important topic in speech translation [15, 16, 17]. Recently, some studies have proposed the direct use of non-streaming models for streaming translation [18, 19, 20, 21], achieving close or even better performance than models requiring streaming-aware training. These works focus on the nearly monotonic cross-attention within appropriately trained encoder-decoder models, enabling control over the time to stop decoding. For example, EDAtt [22] stops decoding when the attention on the last few audio frames reaches a certain threshold, while AlignAtt [19] focuses on the position of maximum attention and stops decoding when it is close enough to the end of the audio. Compared to the Local Agreement method, this type of method requires less computation and has better control over decoding latency. Since Whisper's cross-attention also shows good temporal alignment, we believe that this type of fine-tuning-free method can also be applied to Whisper's streaming inference.

However, a problem with these methods is that cross-attention primarily provides information from the decoder. As shown in Figure 1, truncated words at chunk boundaries can lead to wrong transcriptions, and relying only on decoder information may not be sufficient to discriminate such cases. To exclude these unreliable transcriptions, information from the encoder is crucial. Recently, Dong et al. [23] and Zhang et al. [24] propose to train an integrate-and-fire (IF) model [25, 26, 27] to detect the word boundaries. Similar methods can be applied to word truncation detection, or in other words, if a word boundary is not detected at the end of a chunk, it is equivalent to a word

† Equal contribution
* Corresponding author

This work was supported by the National Natural Science Foundation of China under Grant No. 62276153.


10.21437/Interspeech.2024-1814

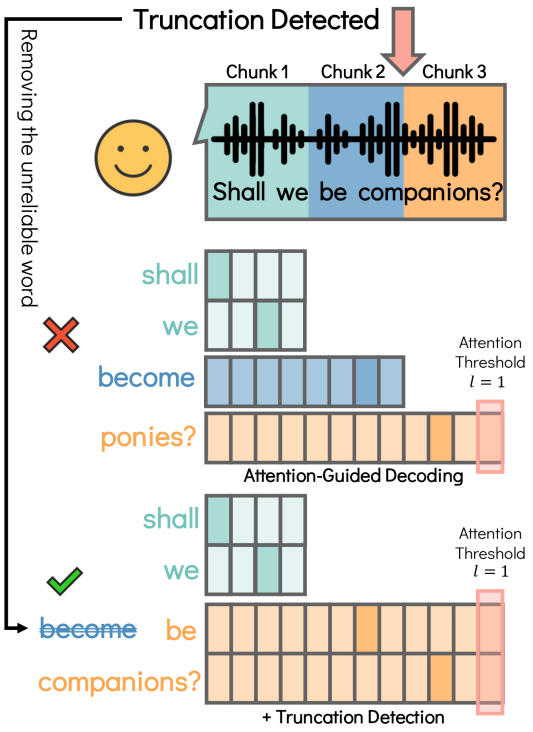


Figure 1: *An overview of our method. Different colours indicate different audio chunks and their corresponding transcriptions. Darker blocks in the cross-attention matrix indicate the audio frames most attended to by the current token. Upper part: Decoding is stopped when the most attended audio frame appears at the chunk boundary. Lower part: The unreliable last word is deleted from the transcription when truncation is detected and the model waits until the next chunk is received.*

truncation.

In this paper, we propose Simul-Whisper, a streaming inference strategy for Whisper that does not require fine-tuning of the pre-trained model[1]. By integrating the encoder and decoder information from both the cross-attention and the truncation detection module, we can track the decoding process, stop decoding at appropriate times, and discard unreliable transcriptions. Our experiments on multiple languages and Whisper architectures demonstrate that the proposed method achieves streaming inference with a chunk size of 1 second with a minimum absolute performance degradation of 0.77%, which significantly outperforms the state-of-the-art Local Agreement baseline.

## 2. Methods

In this section, we introduce our method for streaming inference using non-streaming Whisper without any fine-tuning. First, we utilise Whisper's cross-attention mechanism to obtain a rough alignment and stop decoding at the appropriate time. Second, to further eliminate unreliable transcriptions at chunk boundaries, we introduce a Truncation Detection Module (TDM) based on the Integrate-and-Fire (IF) mechanism.

[1]Available at https://github.com/backspacetg/simul_whisper

### 2.1. Attention-Guided Decoding Policy

Whisper has various model architectures, all of which consist of a 2-layer CNN, along with a multi-layer Transformer encoder and decoder. The encoder-decoder attention is calculated by

$$Q = X_t W_Q^i \tag{1}$$

$$K = X_a W_K^i \tag{2}$$

$$S^i = \text{softmax}(\frac{QK^T}{\sqrt{d_K}}), \tag{3}$$

where $X_a \in \mathbb{R}^{N_a \times D_a}$ is the encoded audio, $X_t$ in the previous outputs of the decoder. $N_a$ and $D_a$ are the length and hidden dimension of the encoded audio, respectively. Similarly, $N_t$ and $D_t$ are those for the the decoder outputs. $S^i$ in $\mathbb{R}^{N_t \times N_a}$ is the output of the $i$th attention head of the multi-head cross-attention module.

During the large-scale weakly supervised training of Whisper, some attention heads in the cross-attention modules exhibit a favourable temporal alignment. To be specific, for the $t$th row $s_t^i$ of the attention matrix, the region where token $t$ shows significant attention to $X_a$ often substantially overlaps with the corresponding audio. These attention heads are manually selected and called alignment heads, and the timestamps generated by Whisper are actually obtained by applying Dynamic Time Warping (DTW) to them. Following the open source code of Whisper[2], we perform further post-processing on the output of the alignment heads. The resulting alignment matrix can be represented as:

$$S = f_m(\sum_{i \in H} s^i), \tag{4}$$

where $f_m$ is a median filter with a window width of 7, $H$ is the set of the alignment heads. To guide the decoding process, we need to find out from the alignment head the position where the current token focuses on in the audio. Inspired by Papi et al. we use the position of the maximum attention as a reference. To be specific, the decoding process will be terminated when

$$N_a - \text{argmax}(S_t) < l \tag{5}$$

to avoid incorrect or repetitive tokens, where $l$ is a predefined threshold. Compared to DTW, this approach treats each token as an independent unit, thereby reducing the cumulative error in the auto-regressive decoding progress. Besides, selecting the maximum value can also reduce the noise caused by the randomness of the model. It is therefore more suitable for tracking the streaming inference process where training and testing are significantly mismatched.

### 2.2. IF-Based Truncation Detection Module

The fixed-length audio chunks provided to the model may contain incomplete speech units, which leads to unreliable transcriptions. Moreover, these errors accumulate and affect the whole sentence. To address these issues, we design a IF-based truncation detection module. If a truncation is not detected during decoding, the resulting token is preserved. Otherwise, the unreliable token is removed and generated when the complete word is received in the next chunk.

The IF neuron continuously receives and integrates signals. When the accumulated signal exceeds a certain threshold, the

[2]https://github.com/openai/whisper

neuron fires, or in other words, generates an output and resets the accumulated value. In our TDM, the signal is generated by mapping the output of Whisper's encoder through a simple linear layer and a sigmoid function. Our training objective is to ensure that the number of firings of the IF neurons matches the number of words in the audio. During inference, a truncation is detected if the IF neuron does not fire at the end of the audio chunk. Given an IF threshold $f$, signal sequence $a \in \mathbb{R}^{N_a}$, accumulated signal $I$, the TDM will record the last position $p$ where IF fires to detect truncation according to the operations below:

**Algorithm 1** truncation detection module

**Input:** $\alpha$
Initialize $I \leftarrow 0, p \leftarrow 0$
**for** $n = 0$ **to** $N_a - 1$ **do**
  Compute $I \leftarrow I + \alpha_n$
  Determine fire where $I \geq f$
  Update $I$:

$$I \leftarrow \begin{cases} I - f, & \text{if fire} \\ I, & \text{otherwise} \end{cases}$$

  Update $p \leftarrow n$
**end for**
Determine truncation where $p \leq N_a - 1$

Since Whisper pads the input to 30 seconds during inference, the last frame of the encoder features is actually the transition point from speech content to silent padding, where the IF neuron always fires. Therefore, to ensure the correctness of the IF results, we discard the last frame of the encoder features.

## 3. Experimental Settings

### 3.1. Data

The proposed method is evaluated on the LibriSpeech dataset [28] and the Multilingual Librispeech (MLS) dataset [29], which consists of seven languages, including Dutch (nl), French (fr), Polish (pl), German (de), Italian (it), Portuguese (pt), and Spanish (es). For the Librispeech dataset, the subsets test-clean, dev-clean, test-other and dev-other are evaluated, and for MLS we only use the test subsets.

### 3.2. Truncation Detection Training Setup

The root mean square error (RMSE) is used as the training objective of the TDM, and our training dataset is the 100-hour Librispeech train-clean subset. The Adam optimizer is used together with warm-up to prevent unstable behaviour in the early stages of training. Our TDM is trained for 10 epochs with a batch size of 4500 MFCC frames, and the first 3 epochs are used for warm-up, where the learning rate increases linearly from 0 to $1 \times 10^{-6}$. Training is performed on a single NVIDIA GeForce RTX 3090 GPU with 24GB of memory.

### 3.3. Inference and Evaluation

The generated transcriptions and ground truth labels are normalized using Whisper's open source text normalizer, and the Python library editdistance [30] is used to evaluate the word error rates. We use a threshold of $l = 12$ frames (240 ms) for attention-guided decoding, and the fire threshold for TDM is 0.999. For the baseline, we reproduce the Local Agreement policy based on its open source code[3] and use the experimental setup described in the original paper, which computes the longest common prefix between $n = 2$ consecutive source chunks. We use the Differentiable Average Lagging (DAL) [31] to evaluate the latency. All inferences are performed on a single NVIDIA GeForce RTX 3090 GPU with 24GB of memory.

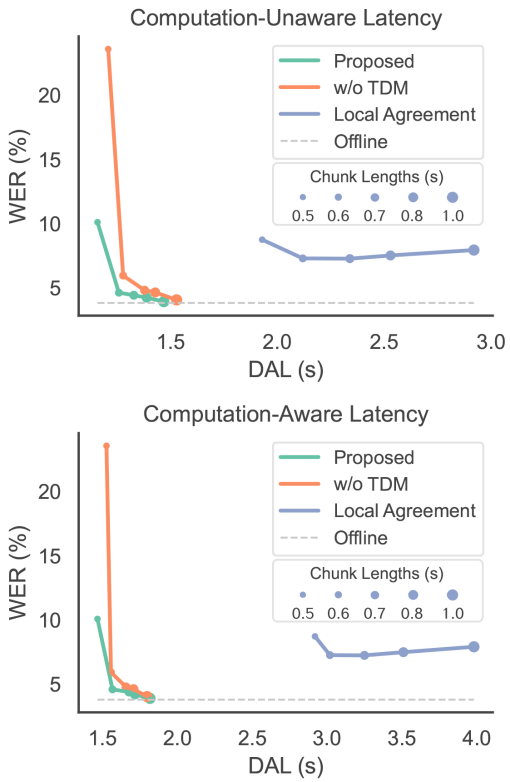


Figure 2: *The WERs at different DALs for Whisper Large-v2, where computation-aware latency takes into account processing time, while computation-unaware latency doesn't. Chunk lengths vary from 0.5 to 1.0 seconds. For the same latency, the proposed method exhibits a significantly lower WER compared to the baseline, and the addition of IF further improves performance.*

## 4. Results

### 4.1. The Word Error Rates of Simul-Whisper

Table 1 shows the WERs for offline decoding and streaming decoding policies on the Librispeech and MLS datasets, where the chunk length for streaming decoding is 1 second. In these experiments, the proposed method achieves streaming decoding with a minimum absolute performance degradation of 0.09%. Generally, the Medium and Large architectures exhibit less performance degradation than the Base and Small architectures. Moreover, the streaming WERs on Librispeech test-clean and dev-clean datasets are usually lower than those in MLS. This observation suggests that the performance degradation in streaming decoding might be related to the model's initial performance. During streaming inference, the input audio is often short and randomly truncated, necessitating the original offline model to be robust.

[3] https://github.com/ufal/whisper_streaming

Table 1: *The WERs (%) of streaming decoding policies on the Librispeech and MLS datasets, with a chunk length of 1 second.* $\bar{\Delta}$ *is the average performance degradation. The proposed method achieves a minimum performance degradation of 0.09%, significantly outperforming the Local Agreement baseline. Using TDM exhibits positive effects on almost all datasets and model architectures.*

| Architectures | Methods | Librispeech | | | | MLS | | | | | | | $\bar{\Delta}$ (↓) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | test-clean | dev-clean | test-other | dev-other | nl | fr | pl | de | it | pt | es | |
| **Base** | Non-streaming | 5.89 | 6.16 | 12.91 | 12.21 | 30.84 | 25.10 | 25.20 | 19.90 | 32.80 | 23.40 | 14.40 | - |
| | Local Agreement | 12.73 | 11.54 | 19.40 | 19.26 | 44.23 | 38.62 | 37.25 | 41.01 | 45.52 | 33.94 | 24.80 | 10.86 |
| | Proposed | 8.07 | 8.55 | 18.58 | 19.16 | 36.47 | 37.96 | 33.46 | 25.84 | 44.52 | 39.56 | 21.08 | **7.68** |
| | w/o TDM | 9.35 | 9.08 | 19.56 | 19.33 | 37.39 | 37.68 | 33.46 | 26.57 | 43.46 | 37.76 | 20.31 | 7.74 |
| **Small** | Non-streaming | 4.18 | 4.41 | 8.89 | 8.40 | 18.17 | 13.74 | 12.10 | 11.51 | 26.55 | 13.84 | 8.91 | - |
| | Local Agreement | 9.29 | 9.48 | 14.20 | 13.83 | 27.69 | 22.17 | 21.39 | 21.10 | 27.00 | 23.68 | 17.12 | 6.93 |
| | Proposed | 5.15 | 5.44 | 11.39 | 11.18 | 25.87 | 18.05 | 16.83 | 17.36 | 27.21 | 16.73 | 10.96 | **3.23** |
| | w/o TDM | 6.23 | 6.14 | 13.13 | 12.65 | 26.60 | 19.54 | 18.79 | 19.04 | 29.64 | 19.63 | 12.73 | 4.86 |
| **Medium** | Non-streaming | 3.80 | 3.78 | 8.50 | 6.84 | 12.40 | 9.43 | 6.78 | 8.24 | 15.51 | 8.89 | 5.93 | - |
| | Local Agreement | 14.61 | 13.46 | 16.24 | 15.77 | 14.76 | 20.51 | 42.11 | 18.00 | 21.57 | 17.55 | 12.08 | 10.59 |
| | Proposed | 3.90 | 4.02 | 8.12 | 7.70 | 15.05 | 10.30 | 11.40 | 10.20 | 18.13 | 11.05 | 7.66 | **1.46** |
| | w/o TDM | 4.36 | 4.26 | 9.23 | 8.74 | 18.37 | 12.15 | 10.43 | 11.35 | 21.03 | 13.40 | 7.90 | 2.83 |
| **Large-v2** | Non-streaming | 3.79 | 3.76 | 7.38 | 6.70 | 9.68 | 6.82 | 5.49 | 6.26 | 13.14 | 6.63 | 4.78 | - |
| | Local Agreement | 7.91 | 6.91 | 11.47 | 11.41 | 18.89 | 15.13 | 13.44 | 14.73 | 20.73 | 16.35 | 13.39 | 6.90 |
| | Proposed | 3.89 | 3.85 | 8.89 | 8.14 | 12.86 | 9.66 | 7.90 | 8.94 | 17.58 | 12.30 | 6.21 | **2.34** |
| | w/o TDM | 4.03 | 3.96 | 9.13 | 8.56 | 15.05 | 11.19 | 9.80 | 13.70 | 18.13 | 12.92 | 8.08 | 3.65 |

We also compare the average performance degradation $\bar{\Delta}$ on all the datasets above. Our proposed method exhibits significantly lower $\bar{\Delta}$ compared to the baseline. We observed that the transcription of the Local Agreement policy often contains insertion or deletion errors at the ends of sentences, which may be related to its context management strategy. Local Agreement retains a variable-length audio context, and when Whisper outputs a delimiter (such as a period), the buffer is truncated from the delimiter based on Whisper's timestamp. Hence, errors may occur if the timestamp is not precise enough. On the other hand, the attention-guided policy does not require precise timestamps. We retain the previous audio segment and its transcription result and insert them into a queue. When the length of the retained context exceeds a fixed threshold, the audio and transcription at the top of the queue are removed. Although the remaining transcriptions slightly precede the current audio, this additional context has less negative effect than incorrectly truncated audio. Furthermore, we use the context as conditional input to the decoder, which can better control the decoding than using it as a prompt in Local Agreement.

Besides, the proposed IF-based truncation detection module has a positive effect on accuracy across almost all model architectures and languages, indicating the generalization ability of our method. For Whisper Medium and Large, Simul-Whisper has comparable performance on Librispeech test-clean and dev-clean datasets, suggesting that the performance degradation in these datasets may be mainly caused by truncated words without TDM.

### 4.2. The Latency of Simul-Whisper

We use Differentiable Average Lagging (DAL) to estimate the latency of streaming policies. DAL is the latency relative to the ideal streaming system averaged over all tokens. Assuming the input audio length is $N_a$ and the number of output tokens is $N_t$, the ideal streaming policy generates a token every $d = N_a/N_t$ seconds. Assuming that token $t$ is generated at time $g(t)$, DAL is calculated as follows:

$$g\prime_d(t) = \begin{cases} g(t) & t = 1 \\ \max(g(t), g\prime_d(t-1) + d) & t > 1 \end{cases} \quad (6)$$

$$\text{DAL} = \frac{1}{N_t} \sum_{t=1}^{N_t} g\prime_d(t) - (t-1)d. \quad (7)$$

By adjusting $g(t)$ to $g\prime(t)$, meaningless negative values can be avoided. We evaluate the computation-unaware and computation-aware latency. For the computation-unaware latency, we ignore the computation time and assume that the output is available immediately after a chunk is received, while for the computation-aware latency, the hardware processing time of the model is included.

Figure 2 shows the latency of the Simul-Whisper and baseline models, with chunk lengths varying from 0.5 to 1.0 seconds. It can be seen that for the same latency, the WER of the proposed model is significantly lower than that of the Local Agreement baseline. Local Agreement has a higher latency because it only generates tokens as the longest common prefix grows, and the latency cannot be directly controlled by the chunk length. On the other hand, the Simul-Whisper's latency is generally 1 to 2 times the chunk length. This is because most tokens are either generated after the current chunk (1-chunk latency) or delayed until the next chunk (2-chunk latency). Finally, although TDM slightly increases computation time, it achieves less performance degradation for the same computation-aware latency, demonstrating its effectiveness.

## 5. Conclusions and Discussions

In this paper, we introduce Simul-Whisper, a streaming decoding policy for Whisper without additional fine-tuning of the pre-trained model. We utilise cross-attention to guide the decoding process and train a truncation detection module to avoid unreliable transcriptions at chunk boundaries. Experiments on multiple datasets show that the proposed method can achieve streaming ASR with an average absolute performance degradation of 1.46% at a chunk size of 1 second.

There are still some problem within Simul-Whisper. Since we prefer a fine-tuning-free method, inputs still need to be padded to 30 seconds to meet Whisper's input requirements, which increases computation-aware latency. In future work, we will explore methods such as self-distillation to address this issue.

# 6. References


[1] A. Mohamed, H.-y. Lee, L. Borgholt, J. D. Havtorn, J. Edin, C. Igel, K. Kirchhoff, S.-W. Li, K. Livescu, L. Maaløe, T. N. Sainath, and S. Watanabe, "Self-supervised speech representation learning: A review," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 6, pp. 1179–1210, 2022.

[2] J. Zhao and W.-Q. Zhang, "Improving automatic speech recognition performance for low-resource languages with self-supervised models," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 6, pp. 1227–1241, 2022.

[3] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. Int. Conf. on Mach. Learn.*, 2023, pp. 28 492–28 518.

[4] S. Radhakrishnan, C.-H. Yang, S. Khan, R. Kumar, N. Kiani, D. Gomez-Cabrero, and J. Tegnér, "Whispering LLaMA: A cross-modal generative error correction framework for speech recognition," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2023, pp. 10 007–10 016.

[5] Y. Gong, S. Khurana, L. Karlinsky, and J. Glass, "Whisper-AT: Noise-robust automatic speech recognizers are also strong general audio event taggers," in *Proc. INTERSPEECH*, 2023, pp. 2798–2802.

[6] R. Ma, A. Liusie, M. J. F. Gales, and K. M. Knill, "Investigating the emergent audio classification ability of ASR foundation models," *arXiv:2311.09363*, 2024.

[7] S. Rathod, M. Charola, and H. A. Patil, "Noise robust whisper features for dysarthric severity-level classification," in *Proc. Pattern Recog. Mach. Intell.*, 2023, pp. 708–715.

[8] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 7829–7833.

[9] Z. Tian, J. Yi, Y. Bai, J. Tao, S. Zhang, and Z. Wen, "Synchronous transformers for end-to-end speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 7884–7888.

[10] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 6783–6787.

[11] Y. Fu, Y. Kang, S. Cao, and L. Ma, "DistillW2v2: A small and streaming wav2vec 2.0 based ASR model," *arXiv:2303.09278*, 2023.

[12] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 3451–3460, 2021.

[13] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "WavLM: Large-scale self-supervised pre-training for full stack speech processing," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 6, pp. 1505–1518, 2022.

[14] D. Macháček, R. Dabre, and O. Bojar, "Turning whisper into real-time transcription system," in *Proc. Int. Joint Conf. on Natural Language Process. Conf. Asia-Pacific Chapter Assoc. for Comput. Linguistics: Syst. Demonstrations*, 2023, pp. 17–24.

[15] A. Bérard, O. Pietquin, C. Servan, and L. Besacier, "Listen and translate: A proof of concept for end-to-end speech-to-text translation," in *Proc. NIPS Workshop End-to-End Learni. Speech Audio Process.*, 2016.

[16] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, "Sequence-to-sequence models can directly translate foreign speech," in *Proc. INTERSPEECH*, 2017, pp. 2625–2629.

[17] X. Ma, J. Pino, J. Cross, L. Puzon, and J. Gu, "Monotonic multihead attention," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 27–41.

[18] S. Papi, M. Gaido, M. Negri, and M. Turchi, "Does simultaneous speech translation need simultaneous models?" in *Findings Assoc. Comput. Linguistics: EMNLP*, 2022, pp. 141–153.

[19] S. Papi, M. Turchi, and M. Negri, "AlignATT: Using attention-based audio-translation alignments as a guide for simultaneous speech translation," *Proc. INTERSPEECH*, pp. 3974–3978, 2023.

[20] M. Ma, L. Huang, H. Xiong, R. Zheng, K. Liu, B. Zheng, C. Zhang, Z. He, H. Liu, X. Li, H. Wu, and H. Wang, "STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3025–3036.

[21] P. Polák, N.-Q. Pham, T. N. Nguyen, D. Liu, C. Mullov, J. Niehues, O. Bojar, and A. Waibel, "CUNI-KIT system for simultaneous speech translation task at IWSLT 2022," in *Proc. Int. Conf. Spoken Language Transl.*, 2022, pp. 277–285.

[22] S. Papi, M. Negri, and M. Turchi, "Attention as a guide for simultaneous speech translation," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2023, pp. 13 340–13 356.

[23] Q. Dong, Y. Zhu, M. Wang, and L. Li, "Learning when to translate for streaming speech," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 680–694.

[24] S. Zhang and Y. Feng, "Information-transport-based policy for simultaneous translation," in *Proc. 2022 Conf. Empirical Methods Natural Language Process.*, 2022, pp. 992–1013.

[25] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain research bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999.

[26] L. Dong and B. Xu, "CIF: Continuous integrate-and-fire for end-to-end speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.* IEEE, 2020, pp. 6079–6083.

[27] C. Yi, S. Zhou, and B. Xu, "Efficiently fusing pretrained acoustic and linguistic encoders for low-resource speech recognition," *IEEE Signal Process. Lett.*, vol. 28, pp. 788–792, 2021.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 5206–5210.

[29] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "MLS: A large-scale multilingual dataset for speech research," in *Proc. Interspeech*, 2020, pp. 2757–2761.

[30] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1091–1095, 2007.

[31] N. Arivazhagan, C. Cherry, W. Macherey, C.-C. Chiu, S. Yavuz, R. Pang, W. Li, and C. Raffel, "Monotonic infinite lookback attention for simultaneous machine translation," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1313–1323.