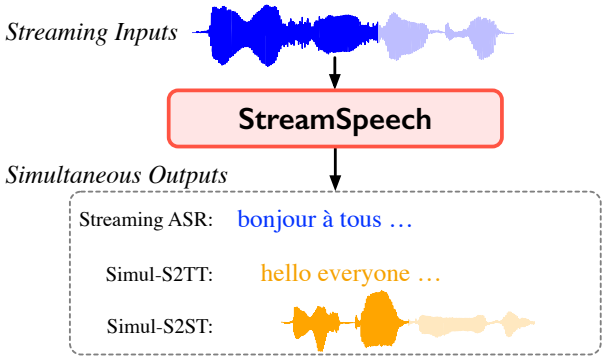
# StreamSpeech: Simultaneous Speech-to-Speech Translation with Multi-task Learning

**Shaolei Zhang**[1,3], **Qingkai Fang**[1,3], **Shoutao Guo**[1,3], **Zhengrui Ma**[1,3], **Min Zhang**[4], **Yang Feng**[1,2,3*]

[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2]Key Laboratory of AI Safety, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences, Beijing, China
[4]School of Future Science and Engineering, Soochow University
zhangshaolei20z@ict.ac.cn, zhangminmt@hotmail.com, fengyang@ict.ac.cn

## Abstract

Simultaneous speech-to-speech translation (Simul-S2ST, a.k.a streaming speech translation) outputs target speech while receiving streaming speech inputs, which is critical for real-time communication. Beyond accomplishing translation between speech, Simul-S2ST requires a policy to control the model to generate corresponding target speech at the opportune moment within speech inputs, thereby posing a double challenge of translation and policy. In this paper, we propose *StreamSpeech*, a direct Simul-S2ST model that jointly learns translation and simultaneous policy in a unified framework of multi-task learning. Adhering to a multi-task learning approach, StreamSpeech can perform offline and simultaneous speech recognition, speech translation and speech synthesis via an "All-in-One" seamless model. Experiments on CVSS benchmark demonstrate that StreamSpeech achieves state-of-the-art performance in both offline S2ST and Simul-S2ST tasks. Besides, StreamSpeech is able to present high-quality intermediate results (i.e., ASR or translation results) during simultaneous translation process, offering a more comprehensive real-time communication experience[1].




"**All in One**" seamless model for offline and simultaneous ASR / Translation / Synthesis

Figure 1: StreamSpeech is an "All in One" seamless model for multiple offline and simultaneous tasks.

## 1 Introduction

Simultaneous speech-to-speech translation (Simul-S2ST), which involves generating target speech while concurrently receiving streaming speech inputs (Salesky et al., 2023; OpenAI, 2024), has become an indispensable technology for low-latency communication in various scenarios, such as international conferences, live broadcasts and online subtitles. To produce high-quality translated speech under low latency, Simul-S2ST requires a policy to determine the optimal moments to start translating within the streaming speech inputs (i.e., READ action) and subsequently generate coherent target speech outputs (i.e., WRITE action) (Gu et al., 2017).

Existing simultaneous translation methods focus on text-to-text (Simul-T2TT) (Ma et al., 2019; Arivazhagan et al., 2019; Zhang and Feng, 2023b) and speech-to-text translation (Simul-S2TT) (Ren et al., 2020; Chen et al., 2021; Zeng et al., 2021; Zhang and Feng, 2023a). Such methods typically require cascading external modules such as speech recognition (ASR) and text-to-speech synthesis (TTS) to accomplish Simul-S2ST. However, this cascaded approach tends to amplify inference errors progressively between modules (Zhang et al., 2022a; Ma et al., 2020b), and also impedes the joint optimization of various modules (Zhang and Feng, 2023c). To address these issues, developing a direct Simul-S2ST model is imperative, particularly given the promising potential exhibited by end-to-end models such as SeamlessM4T (Communication et al., 2023b) and GPT-4o (OpenAI, 2024).

Direct speech-to-speech translation (S2ST) is already highly challenging, and the goal of accomplishing it simultaneously (Simul-S2ST) further

*Corresponding author: Yang Feng
[1]Code: https://github.com/ictnlp/StreamSpeech
Project Page: https://ictnlp.github.io/StreamSpeech-site/



*Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8964–8986

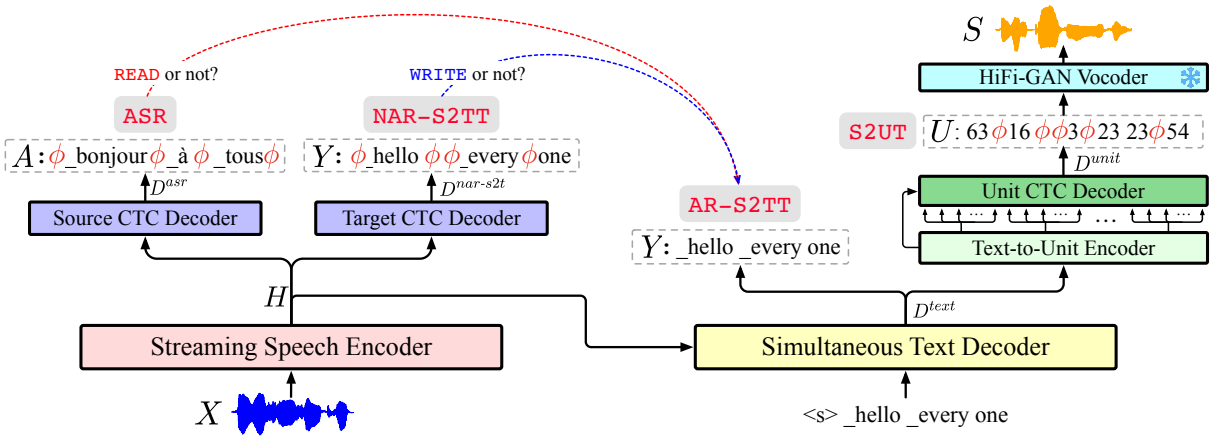



Figure 2: StreamSpeech employs two-pass architecture that first converts source speech into target text hidden states $D^{text}$ (autoregressive speech-to-text translation, AR-S2TT) and then generates target speech via non-autoregressive text-to-unit generation. The source/target/unit CTC decoders are introduced to learn alignments via multiple tasks of speech recognition (ASR), non-autoregressive speech-to-text translation (NAR-S2TT) and speech-to-unit translation (S2UT), accordingly guiding StreamSpeech when to start recognizing, translating and synthesizing.

exacerbates the difficulty. For translation, speech involves more diverse representation due to additional features such as timbre and intonation (Jia et al., 2022a), which renders directly translating source speech to target speech challenging. In simultaneous scenarios, beyond translation, the model additionally requires a policy to identify the appropriate translating moments, which is non-trivial to directly accomplish due to the continuous nature and uncertain duration of speech (Zhang and Feng, 2023a). Therefore, Simul-S2ST faces the double challenges of translation and policy.

To address the challenges of translation and policy, we aim to introduce textual information of both source and target speech to guide Simul-S2ST, which can not only provide intermediate supervision for translation but also guide the policy by establishing an alignment between source and target speech with text as a bridge. Specifically, a reasonable policy should control the model to wait until recognizing the presence of text in the received speech (READ), facilitated by the alignment between source speech and source text. Subsequently, the model should generate target speech corresponding to inputs (WRITE), which can be guided by the alignments from the source speech to target text and from the target text to target speech.

Given the pivotal role of text in both translation and alignment-guided policy, we propose *StreamSpeech*, a direct Simul-S2ST model that jointly learns translation and policy in a unified framework of multi-task learning. StreamSpeech employs the advanced two-pass architecture (Inaguma et al., 2023; Jia et al., 2022a), which first translates source speech into target text hidden states, and then converts the text hidden states into target speech. Furthermore, we introduce multiple connectionist temporal classification (CTC) (Graves et al., 2006) decoders and optimize them via auxiliary tasks of ASR and S2TT, thereby providing intermediate supervision for translation and meanwhile learning alignments to guide policy. All modules in StreamSpeech are jointly optimized through multi-task learning, facilitating jointly learning of translation and policy. Experiments show that StreamSpeech exhibits adaptability to different latency and achieves state-of-the-art performance on both offline S2ST and Simul-S2ST tasks.

## 2 Background

**Speech-to-Speech Translation (S2ST)** The corpus we used for speech-to-speech translation (S2ST) task is denoted as quadruple: $\mathcal{D} = \{(X, A, Y, S)\}$, where $X = (x_1, \cdots, x_{|X|})$ is the source speech, $A = (a_1, \cdots, a_{|A|})$ is the transcribed text of source speech, $Y = (y_1, \cdots, y_{|Y|})$ is the target text, $S = (s_1, \cdots, s_{|S|})$ is the target speech. The current mainstream methods for S2ST (Inaguma et al., 2023) extract a discrete unit sequence $U = (u_1, \cdots, u_{|U|})$ from the target speech, and employ a two-pass architecture, where both the first and second passes use autoregressive encoder-decoder. The first pass transforms the source speech to target text hidden states, and the second pass generates the discrete unit sequence based on the text hidden states, followed by a pre-trained

unit-based HiFi-GAN vocoder (Kong et al., 2020) for target speech synthesis. In addition to the primary speech-to-unit translation (S2UT, $X \rightarrow U$), an auxiliary speech-to-text translation task (S2TT, $X \rightarrow Y$) is introduced to provide supervision.

**Connectionist Temporal Classification (CTC)** (Graves et al., 2006) CTC is a technique used to model alignment between two sequences of unequal lengths. For a longer input sequence $\mathcal{X}$, CTC decoder generates a same-length sequence $\mathcal{Z}$ containing repeated and blank tokens $\phi$, which is subsequently shortened by merging consecutively repeated tokens and removing blank tokens $\phi$ via collapsing function $\Pi(\cdot)$. During training, given the ground-truth sequence $\mathcal{Y}$, CTC loss is calculated on all sequences $\mathcal{Z}$ that can be reduced to $\mathcal{Y}$ via the collapsing function:

$$\mathrm{CTC}(\mathcal{X}, \mathcal{Y}) = -\log \sum_{\mathcal{Z} \in \Pi^{-1}(\mathcal{Y})} p(\mathcal{Z} \mid \mathcal{X}). \quad (1)$$

## 3 StreamSpeech

### 3.1 Architecture

The overall architecture of StreamSpeech is illustrated in Figure 2. StreamSpeech consists of three parts: streaming speech encoder, simultaneous text decoder and synchronized text-to-unit generation module. Multiple CTC decoders are introduced to learn the alignments through auxiliary tasks and accordingly guide the policy.

**Streaming Speech Encoder** Conformer architecture exhibits remarkable advantages in speech modeling by stacking attention modules and convolutional modules (Gulati et al., 2020), while it struggles to model the streaming speech inputs, primarily due to the bi-directional self-attention and convolutional operations involving the entire sequence's receptive field. To this end, we propose *chunk-based Conformer*, aiming to endow the Conformer architecture with the capability to encode streaming inputs while retaining the bi-directional encoding within local chunk.

Figure 3 shows the architecture of chunk-based Conformer. First of all, the raw speech inputs are converted to speech features (we use filterbank features (Povey et al., 2011) in our work), where each speech feature typically corresponds to a $40ms$ duration. Chunk-based Conformer divides the streaming speech into chunks, each containing $C$ speech features, where $C$ is a hyperparameter controlling the chunk size. In the chunk-based Conformer,

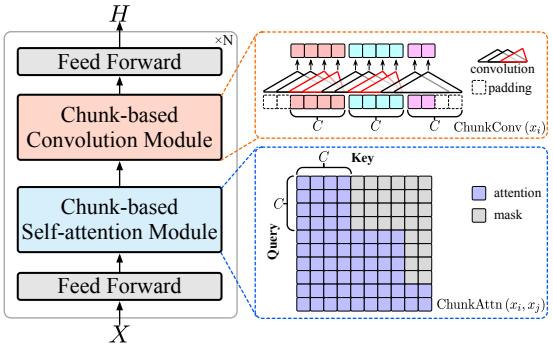



Figure 3: Architecture of chunk-based Conformer.

self-attention and convolution operations are both bidirectional within a chunk and unidirectional between chunks, thereby handling the streaming inputs. For chunk-based self-attention, feature $x_i$ pays attention to the features $x_j$ that are located in the same and previous chunks, calculated as:

$$\mathrm{ChunkAttn}(x_i, x_j) = \begin{cases} \mathrm{Attn}(x_i, x_j) & \text{if } j \leq \lceil \frac{i}{C} \rceil \times C \\ 0 & \text{otherwise} \end{cases}, \quad (2)$$

where $\mathrm{Attn}(x_i, x_j)$ is standard multi-head attention (Vaswani et al., 2017), and $\lceil \cdot \rceil$ is ceiling operation. For chunk-based convolution, the convolution operation with kernel size $k$ is truncated at the upper bound of the chunk, calculated as:

$$\mathrm{ChunkConv}(x_i) = \mathrm{Conv}\left(x_{i-\frac{k-1}{2}}, \cdots, x_i, \cdots, x_{\min(i+\frac{k-1}{2}, \lceil \frac{i}{C} \rceil \times C)}\right). \quad (3)$$

where $\lceil \frac{i}{C} \rceil \times C$ is the upper bound of the chunk that $x_i$ is located in. In implementation, chunk-based convolution can be computed in parallel through a mask operation (mask out those truncated positions). Through the streaming encoder, the source speech hidden states are calculated, denoted as $H = (h_1, \cdots, h_{|H|})$. With chunk-based Conformer, the streaming speech encoder not only fulfills the need for streaming encoding but also conducts local bi-directional encoding of speech.

**Simultaneous Text Decoder** After streaming encoder, text decoder simultaneously generates target text $Y$ by attending the source speech hidden states $H$. To achieve this, StreamSpeech requires a policy to decide when to generate each target token (i.e., how many speech states can the decoder attend to.). A reasonable policy should ensure that the model waits until recognizing the source text in the source speech (READ), and then generates the corresponding target text (WRITE).

To this end, we aim to align the source and target text to the speech inputs, thereby guiding "READ or not" and "WRITE or not" respectively. Considering the length difference between speech and text sequences, we align them via CTC decoder (refer to Sec.2). Specifically, we introduce a source CTC decoder $\texttt{CTCDec}^{\texttt{A}}(\cdot)$ and a target CTC decoder $\texttt{CTCDec}^{\texttt{Y}}(\cdot)$ at the top of the streaming speech encoder to generate source and target text:

$$D^{asr} = \texttt{CTCDec}^{\texttt{A}}(H), \tag{4}$$

$$D^{nar\text{-}s2tt} = \texttt{CTCDec}^{\texttt{Y}}(H), \tag{5}$$

and optimize them through the auxiliary tasks of speech recognition (ASR, $X \rightarrow A$) and non-autoregressive speech-to-text translation (NAR-S2TT, $X \rightarrow Y$), via CTC loss respectively:

$$\mathcal{L}_{asr} = \mathrm{CTC}(D^{asr}, A), \tag{6}$$

$$\mathcal{L}_{nar\text{-}s2tt} = \mathrm{CTC}(D^{nar\text{-}s2tt}, Y). \tag{7}$$

With CTC decoders, the source and target text are aligned to source speech. Accordingly, StreamSpeech starts translating upon the source CTC decoder recognizing a new source token from source speech, and then autoregressively generates target tokens that align to the received speech within target CTC decoder[2]. Therefore, we calculate the number of source tokens and target tokens aligned to the current speech inputs $X_{\leq j}$, denoted as $\mathcal{N}_j^{asr}$ and $\mathcal{N}_j^{nar\text{-}s2t}$, respectively. Note that during training, we calculate the expected number of tokens contained in the CTC sequence, where the specific calculation is introduced in Appendix A.

Given $\mathcal{N}_j^{asr}$ and $\mathcal{N}_j^{nar\text{-}s2tt}$, StreamSpeech autoregressively generates target token $y_i$ after receiving speech $X_{\leq g(i)}$, where $g(i)$ is defined as:

$$g(i) = \underset{\{j \,|\, \mathcal{N}_{j-1}^{asr} < \mathcal{N}_j^{asr}\}}{\operatorname{argmin}} \left(\mathcal{N}_j^{nar\text{-}s2tt} \geq i\right). \tag{8}$$

$\mathcal{N}_{j-1}^{asr} < \mathcal{N}_j^{asr}$ ensures that StreamSpeech starts translating when a new source token is recognized, and $(\mathcal{N}_j^{nar\text{-}s2tt} \geq i)$ ensures that StreamSpeech generates those target tokens that align to the received speech. Based on the policy guided by the alignments derived from ASR and NAR-S2TT, simultaneous text decoder generates $y_i$ after receiving speech $X_{\leq g(i)}$, and optimized via cross-entropy loss on autoregressive speech-to-text translation (AR-S2TT, $X \rightarrow Y$):

$$\mathcal{L}_{ar\text{-}s2tt} = -\frac{1}{|Y|}\sum_{i=1}^{|Y|} \log p\left(y_i \mid X_{\leq g(i)}, Y_{<i}\right). \tag{9}$$

**Non-autoregressive Text-to-Unit Generation** To synchronously generate the corresponding unit for the currently generated text, StreamSpeech employs a non-autoregressive text-to-unit (T2U) architecture (Gu et al., 2018), comprising a T2U encoder and a unit CTC decoder. T2U encoder takes the hidden state $D^{text}$ from the simultaneous text decoder as inputs. For the unit CTC decoder, considering that unit sequences $U$ are often longer than text sequences $Y$, we upsample the T2U encoder outputs $r$ times as the decoder inputs, where the $i^{th}$ input corresponds to $D^{text}_{\lceil i/r \rceil}$. Then unit CTC decoder generates the unit sequence $U$ non-autoregressively by attending to those T2U encoder outputs located before $D^{text}_{\lceil i/r \rceil}$. Formally, the output $D^{unit}$ of unit CTC decoder $\texttt{CTCDec}^{\texttt{U}}$ is calculated as:

$$D_i^{unit} = \texttt{CTCDec}^{\texttt{U}}\left(D^{text}_{\leq \lceil i/r \rceil}\right). \tag{10}$$

NAR T2U generation is optimized on speech-to-unit translation task (S2UT, $S \rightarrow U$) via CTC loss:

$$\mathcal{L}_{s2ut} = \mathrm{CTC}(D^{unit}, U). \tag{11}$$

Finally, a unit-based HiFi-GAN vocoder (Kong et al., 2020) is used to synthesize target speech based on the generated units. Note that the HiFi-GAN vocoder is often pre-trained and frozen.

### 3.2 Training

All tasks involved in StreamSpeech are jointly optimized via multi-task learning in an end-to-end manner, and the total training objective $\mathcal{L}$ encompasses the losses of S2UT, AR-S2TT, ASR, and NAR-S2TT tasks:

$$\mathcal{L} = \mathcal{L}_{s2ut} + \mathcal{L}_{ar\text{-}s2tt} + \mathcal{L}_{asr} + \mathcal{L}_{nar\text{-}s2tt}. \tag{12}$$

Multi-task learning effectively integrates the learning of simultaneous policy and translation into a unify framework. Besides, the high-quality intermediate results of auxiliary tasks, such as ASR and AR-S2TT, can also be displayed to users during inference as supplementary products.

**Multi-chunk Training** During inference, Simul-S2ST may face different latency requirements, and training multiple models for every latency is expensive (Elbayad et al., 2020; Zhang and Feng, 2021b).

[2]NAR-S2TT can achieve well 1-gram token accuracy, but its translations are often less smooth compared to AR-S2TT. Therefore, StreamSpeech adopts NAR-S2TT to capture alignment and guide the policy, while still leveraging AR-S2TT to generate target tokens for better translation quality.

**Algorithm 1:** Inference of StreamSpeech

**Input :** streaming speech inputs $X$, chunk size $C$, current received speech $\widehat{X}$
**Output :** target speech outputs $S$

1 **while** $|\widehat{X}| \leq |X|$ **do**
2  generate ASR results $\widehat{A}$, with Eq.(5);
3  generate NAR-S2TT results $\widehat{Y}$, with Eq.(5);
4  **if** $|\widehat{A}| > |A|$ **and** $|\widehat{Y}| > |Y|$ **then** // WRITE
5  $A = \widehat{A}$;
6  **while** $|Y| < |\widehat{Y}|$ **and** $Y_{-1} \neq$ <*eos*> **do**
7  generate target token $y$, with Eq.(9);
8  $Y$.append($y$)
9  **end**
10  generate units $U$ of $Y$, with Eq.(10);
11  $S = \text{Vocoder}(U)$;
12  // output new generated speech
13  **else** // READ
14  wait for next speech chunk;
15  $\widehat{X}$.append($X_{|\widehat{X}|:|\widehat{X}|+C}$);
16  **end**
17 **end**

To this end, we introduce multi-chunk training to improve the performance of StreamSpeech across various latency levels. In multi-chunk training, chunk size $C$ of streaming speech encoder is not fixed, but randomly sampled from a uniform distribution for 1 to $|X|$, expressed as $C \sim \mathcal{U}(1, |X|)$, where $c = |X|$ refers to offline S2ST. With multi-chunk training, a single StreamSpeech model can cater to different latency requirements.

### 3.3 Inference

Algorithm 1 illustrates the inference policy of StreamSpeech. During inference, StreamSpeech processes streaming speech inputs based on the set chunk size $C$, where each speech feature typically corresponds to $40ms$ duration (e.g., $C = 8$ means encoding speech inputs every $C \times 40 = 320ms$). Then StreamSpeech decodes the source tokens $\widehat{A}$ and target tokens $\widehat{Y}$ associated with the currently received speech $\widehat{X}$ through the CTC decoders for ASR and NAR-S2TT tasks. In cases where new source token is recognized, and the count of aligned target tokens surpasses the previously generated target tokens (line 5), StreamSpeech autoregressively generates the target tokens (line 7-10) and synchronously generates the corresponding units (line 11) and synthesizes the target speech (line 12); otherwise StreamSpeech waits for the next speech chunk of size $C$. Due to the proposed multi-chunk training, StreamSpeech can control the latency by adjusting chunk size $C$ during inference, where the smaller $C$ will lead to lower latency.

## 4 Experiments

### 4.1 Experimental Setup

**Datasets** We conduct experiments on CVSS-C benchmark (Jia et al., 2022b), which is a large-scale S2ST corpus derived from the CoVoST 2 speech-to-text translation corpus (Wang et al., 2020) by synthesizing the target speech using a state-of-the-art TTS system. We evaluate StreamSpeech on CVSS-C French→English (Fr→En), Spanish→English (Es→En) and German→English (De→En).

**Pre-processing** Following Inaguma et al. (2023), we convert the source speech to 16000Hz and generate target speech with 22050Hz. For source speech, we compute 80-dimensional mel-filterbank features (Povey et al., 2011) and apply global-level cepstral mean-variance normalization, where each speech feature corresponds to $40ms$ duration. For target speech, we extract the discrete units via mHuBERT[3] (Popuri et al., 2022), and synthesize target speech through a pre-trained unit-based HiFi-GAN vocoder[4] (Kong et al., 2020). For source and target text, we use SentencePiece (Kudo and Richardson, 2018) to generate a unigram vocabulary of size 6000, respectively.

### 4.2 Systems Settings

Since StreamSpeech can be applied to simultaneous and offline S2ST, we compare StreamSpeech with offline S2ST and Simul-S2ST models.

Offline S2ST baselines include **S2UT** (Lee et al., 2022), **Translatotron** (Jia et al., 2019), **Translatotron 2** (Jia et al., 2022a), **DASpeech** (Fang et al., 2023) and **UnitY** (Inaguma et al., 2023), where UnitY is the state-of-the-art offline S2ST model and is the basic framework of seamlessM4T (Communication et al., 2023a). Refer to Appendix C for detailed introduction to these offline baselines.

Simul-S2ST baselines include:

**Wait-k** (Ma et al., 2019) Wait-k policy first waits for $k$×$320ms$ of speech, and then generates a target word every $320ms$ (Ma et al., 2020b). We apply wait-k policy on UnitY, where the first pass adopts wait-k policy, and then the second pass generates units until <eos> token.

**ASR+HMT+TTS (cascaded)** (Zhang and Feng, 2023b) Hidden Markov Transformer[5] (HMT) is

[3] https://dl.fbaipublicfiles.com/hubert/mhubert_base_vp_en_es_fr_it3.pt

[4] https://dl.fbaipublicfiles.com/fairseq/speech_to_speech/vocoder/code_hifigan/mhubert_vp_en_es_fr_it3_400k_layer11_km1000_lj

[5] https://github.com/ictnlp/HMT

| Models | #Param. | Fr→En | | Es→En | | De→En | | Average | |
|---|---|---|---|---|---|---|---|---|---|
| | | greedy | beam10 | greedy | beam10 | greedy | beam10 | greedy | beam10 |
| **Ground Truth** | - | 84.52 | | 88.54 | | 75.53 | | 82.86 | |
| **S2UT** | 73M | 20.91 | 22.23 | 16.94 | 18.53 | 2.46 | 2.99 | 13.44 | 14.58 |
| **Translatotron** | 79M | 16.96 | / | 8.72 | / | 1.97 | / | 9.22 | / |
| **Translatotron 2** | 87M | 25.49 | 26.07 | 22.35 | 22.93 | 16.24 | 16.91 | 21.36 | 21.97 |
| **DASpeech** | 93M | 25.03 | / | 21.37 | / | 16.14 | / | 20.85 | / |
| **UnitY** | 67M | 26.90 | 27.77 | 23.93 | 24.95 | 18.19 | 18.74 | 23.01 | 23.82 |
| **StreamSpeech** | 70M | **27.58**$^{**}$ | **28.45**$^{**}$ | **26.16**$^{**}$ | **27.25**$^{**}$ | **19.72**$^{**}$ | **20.93**$^{**}$ | **24.49** | **25.54** |

Table 1: Offline S2TT results (ASR-BLEU) on CVSS-C Fr→En, Es→En, De→En test sets. We report the results under greedy and beam=10 decoding, where Translatotron only supports greedy decoding and DASpeech uses Viterbi decoding. $^{**}$ means the improvements over the SOTA UnitY are statistically significant ($p < 0.01$).

the state-of-the-art simultaneous text-to-text translation model. We train the streaming ASR and real-time TTS model and add them before and after HMT to form a cascaded Simul-S2ST system.

**DiSeg+TTS (cascaded)** (Zhang and Feng, 2023a) Differentiable segmentation[6] (DiSeg) is the state-of-the-art simultaneous speech-to-text translation model. We also add real-time TTS model after DiSeg to form a cascaded Simul-S2ST system.

**StreamSpeech** Our direct Simul-S2ST model.

All implementations are adapted from Fairseq Library (Ott et al., 2019). StreamSpeech uses basically the same settings as UnitY (Inaguma et al., 2023), and the introduced CTC decoder consists of only a fully connected layer. Other model configurations and training details are reported in Appendix H. The only hyperparameter that needs to be set in StreamSpeech is the upsampling rate $r$ in NAR T2U generation, where we set $r = 25$ based on validation in Appendix D. For cascaded systems, the streaming ASR and real-time TTS modules use the streaming encoder and non-autoregressive text-to-unit module, identical to those used in StreamSpeech, for a fair comparison.

### 4.3 Evaluation

We apply SimulEval[7] (Ma et al., 2020a) to evaluate the Simul-S2ST from both quality and latency.

**Quality** We evaluate S2ST quality using ASR-BLEU toolkit[8], which first transcribes the translated speech into text using a pre-trained ASR model and then calculates the SacreBLEU (Post, 2018) score with reference. We also use BLASER 2.0 to assess the generated speech's quality and the results are reported in Appendix E and J.

[6] https://github.com/ictnlp/DiSeg

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

[8] https://github.com/facebookresearch/fairseq/tree/ust/examples/speech_to_speech/asr_bleu

| Models | Fr→En | | Es→En | | De→En | |
|---|---|---|---|---|---|---|
| | ASR-BLEU | Speedup | ASR-BLEU | Speedup | ASR-BLEU | Speedup |
| **UnitY** | 27.77 | 1.0× | 24.95 | 1.0× | 18.74 | 1.0× |
| **StreamSpeech** | **28.45** | **3.6×** | **27.25** | **4.5×** | **20.93** | **4.5×** |

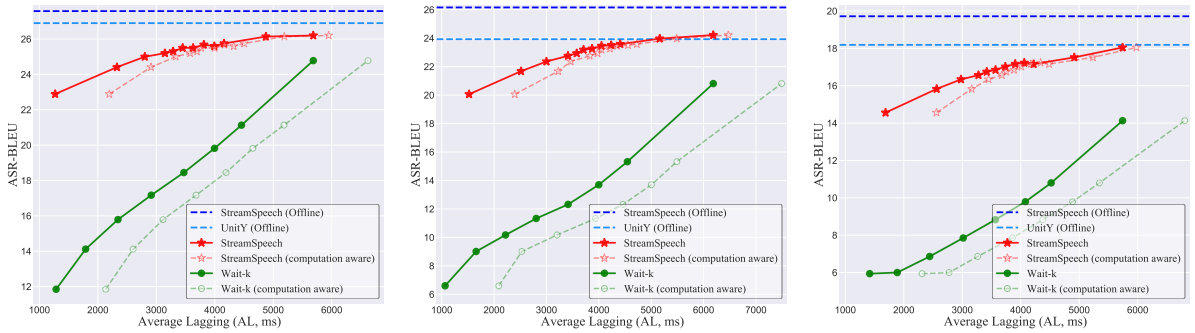
Table 2: Speedup of StreamSpeech.

**Latency** We use Average Lagging (AL) (Ma et al., 2020b) to evaluate the latency, where AL measures the average duration ($ms$) that outputs lag behind inputs. We also measure the computation-aware latency, which includes the computational time of the model. The computation-aware latency is evaluated on 1 NVIDIA RTX 3090 GPU with batch-size=1. More latency metrics are reported in Appendix I to show latency performance.

### 4.4 Main Results

We conduct experiments in both offline S2ST and Simul-S2ST tasks.

**Offline S2ST** Table 1 reports the performance of StreamSpeech in offline S2ST, where StreamSpeech outperforms the state-of-the-art UnitY with an average improvement of 1.5 BLEU. StreamSpeech uses two-pass architecture and achieves significant improvements over S2UT and Translatotron, which use one-pass architecture. For two-pass architecture, DASpeech employs NAR architecture in both first and second passes (Fang et al., 2023), while UnitY uses AR architecture in two passes (Inaguma et al., 2023). StreamSpeech uses AR architecture in S2TT task (first pass) that involves more reordering and context dependence, and NAR architecture in T2U task (second pass) that is basically monotonically aligned. This effectively mitigates the impact of the NAR architecture on modeling capabilities and meanwhile captures the alignment between text and unit. Overall, multi-




(a) Fr→En (b) Es→En (c) De→En

Figure 4: Simul-S2ST results (quality against latency) on CVSS-C Fr→En, Es→En, De→En test sets. The hollow points represent computation-aware latency, which includes the inference time consumed by the model. Some simultaneous outputs of StreamSpeech can be heard at https://ictnlp.github.io/StreamSpeech-site/.

| Models | Tasks | | | | ASR | NAR-S2TT | | AR-S2TT | | S2UT | S2ST |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | ASR | NAR-S2TT | AR-S2TT | S2UT | WER↓ | BLEU↑ | ACC↑ | BLEU↑ | ACC↑ | BLEU↑ | ASR-BLEU↑ |
| **UnitY** | ✘ | ✘ | ✔ | ✔ | / | / | | 31.31 | 61.0 | **33.47** | 27.77 |
| **StreamSpeech** | ✘ | ✘ | ✔ | ✔ | / | / | | 31.20 | 61.5 | 31.37 | 27.47 |
| | ✘ | ✔ | ✔ | ✔ | / | 22.95 | 59.9 | 31.56 | 61.1 | 31.15 | 27.73 |
| | ✔ | ✘ | ✔ | ✔ | 20.70 | / | | 32.28 | 62.3 | 31.42 | 28.18 |
| | ✔ | ✔ | ✔ | ✔ | **20.55** | **23.82** | **60.9** | **32.60** | **62.4** | 31.72 | **28.45** |

Table 3: Ablation study of multi-task learning on offline S2ST, evaluated on CVSS-C Fr→En test set. We report word error rate (WER) for ASR task, BLEU score and 1-gram accuracy (ACC) for NAR-S2TT and AR-S2TT tasks, BLEU score (computes on unit sequences) for S2UT task, and ASR-BLEU score for S2ST task.

task learning not only guides the policy, but also provides intermediate supervision for translation, yielding superior offline S2ST performance.

**Speedup of StreamSpeech** To explore the inference efficiency of StreamSpeech, we report the speedup of StreamSpeech compared to UnitY in Table 2. In the two-pass architecture, StreamSpeech employs an autoregressive structure in the first pass for translation and a non-autoregressive structure in the second pass for speech synthesis (where the sequences are longer but monotonically aligned). This AR+NAR two-pass architecture brings significant speedup while maintaining translation quality.

**Simul-S2ST** Figure 4 shows the Simul-S2ST performance of StreamSpeech, where StreamSpeech outperforms Wait-k under all latency, particularly exhibiting a roughly 10 BLEU improvement under low latency. Wait-k stands as the most widely used policy and achieves good performance on simultaneous T2TT and S2TT (Ma et al., 2019, 2020b). For the Simul-S2ST task where the source and target sequences are both continuous speech, StreamSpeech's policy derived from alignments enables the model to translate at more appropriate moments and generate coherent target speech, resulting in significant advantages. Moreover, concerning computation-aware latency, StreamSpeech introduces only a marginal increase in parameters, thus avoiding notable inference overhead.

**Direct Simul-S2ST v.s. Cascaded Simul-S2ST** Figure 5 presents a comparison between the direct and cascaded Simul-S2ST models, evaluated on the CVSS-C Fr→En test set. The results suggest a general superiority of the direct model over the cascaded systems. Specifically, when we decompose the direct StreamSpeech into two modules "S2TT+TTS", error accumulation leads to a 1 BLEU decrease under low latency, even with the same policy. Furthermore, compared to the cascaded system comprising state-of-the-art HMT and DiSeg, StreamSpeech demonstrates a significant advantage, underscoring the superiority of direct StreamSpeech in Simul-S2ST task.

## 5 Analyses

### 5.1 Effect of Multi-task Learning

StreamSpeech jointly optimizes S2UT, AR-S2TT, ASR, and NAR-S2TT tasks through multi-task learning. To verify the effect of multi-task learning, we conduct an ablation study of auxiliary tasks on

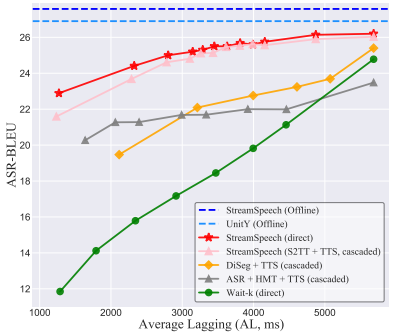



Figure 5: Comparison of direct and cascaded Simul-S2ST systems.

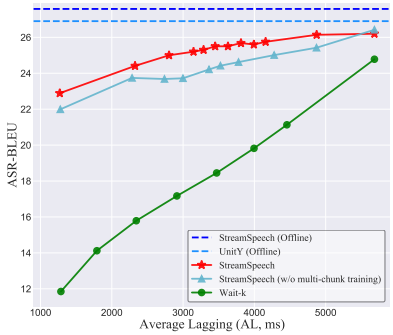



Figure 6: Effect of multi-chunk training in StreamSpeech.

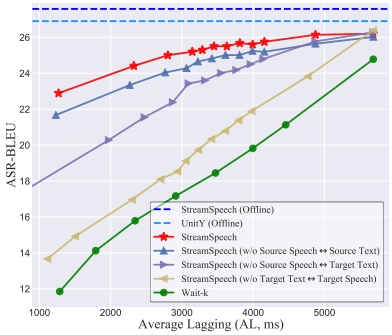



Figure 7: Ablation study on alignments in policy.

| Train \ Test | $C$=8 | $C$=16 | $C$=32 | $C$=64 | $C$=$\infty$ |
|---|---|---|---|---|---|
| $C = 8$ | 24.91 | 24.72 | 25.03 | 24.82 | 23.37 |
| $C = 16$ | 24.18 | 25.64 | 25.75 | 25.62 | 24.76 |
| $C = 32$ | 23.06 | 24.69 | 25.82 | 25.85 | 25.75 |
| $C = 64$ | 19.55 | 22.77 | 24.63 | 25.94 | 26.41 |
| $C = \infty$ | 1.42 | 7.12 | 14.58 | 21.76 | **26.90** |
| Multi-Chunk | **25.34** | **25.97** | **26.31** | **26.61** | 26.47 |

Table 4: Offline S2ST results on various chunk size $C$ of streaming encoder during training and testing.

CVSS-C Fr→En offline S2ST.

As reported in Table 3, the introduction of auxiliary tasks effectively improves the performance of S2ST. Multi-task learning offers staged intermediate supervision for each module within StreamSpeech (Tang et al., 2021a,b), thereby enhancing overall performance. Furthermore, it is noteworthy that NAR-S2TT exhibits a notable gap in BLEU scores compared to AR-S2TT, while the 1-gram accuracy shows minimal differences. This highlights the rationale behind utilizing NAR-S2TT for aligning source speech and target text and employing AR-S2TT for translation.

### 5.2 Superiority of Multi-chunk Training

To enhance the performance of StreamSpeech under various latency, we propose multi-chunk training. Figure 6 illustrates the Simul-S2ST performance on Fr→En test set when employing multi-chunk training and training multiple separate models for different latency. The results indicate that multi-chunk training performs better under all latency. More importantly, multi-chunk training enables StreamSpeech to handle Simul-S2ST under various latency conditions using just one model.

**Chunk-based Conformer** To further evaluate the impact of multi-chunk training on the modeling capabilities of chunk-based Conformer, we conduct experiments by training StreamSpeech with various chunk sizes $C$ and testing them with different test chunk sizes in offline S2ST. The results are reported in Table 4, evaluated on CVSS-C Fr→En test set. The results indicate that models trained with a single chunk size often excel only at a particular test chunk size and struggle to adapt to others. Multi-chunk training equips StreamSpeech with the capability to adapt to different chunk sizes, thereby enabling it to handle S2ST under various latency conditions using a single model. Notably, multi-chunk training also demonstrates superior performance at smaller chunk sizes, which in line with previous findings suggesting that incorporating future information during training has a positive improvement (Ma et al., 2019; Zhang et al., 2021; Zhang and Feng, 2022d; Guo et al., 2024b).

### 5.3 Analysis on StreamSpeech Policy

StreamSpeech models alignments between source speech and source/target text, target text and target speech, thereby allowing for the adaptive decision of READ/WRITE actions. To assess the significance of these three alignments, we present the CVSS-C Fr→En performance when removing one of them (refer to Appendix B for detailed introduction of ablation settings) in Figure 7.

The results underscore the pivotal role of modeling the alignment between target text and target speech through NAR text-to-unit module, as the number of units corresponding to text is often diverse, and the proposed unit CTC decoder effectively addresses this issue. Besides, the alignment between source speech and source/target text ensures StreamSpeech starts translating at appropriate moments and generates a reasonable number of target tokens, where removing any of these alignment components results in performance degradation.

| Models | #Parm. | AL ($ms$)↓ | WER↓ |
|---|---|---|---|
| **Wav2Vec2-large** | 315M | 5684.38 | 26.17 |
| **Whisper-base** | 74M | 5684.38 | 38.04 |
| **StreamSpeech** | 70M (33M used) | 109.127 | 25.46 |
| | | 267.891 | 25.54 |
| | | 431.652 | 25.20 |
| | | 757.989 | 24.67 |

Table 5: Streaming ASR results on Fr→En test set.

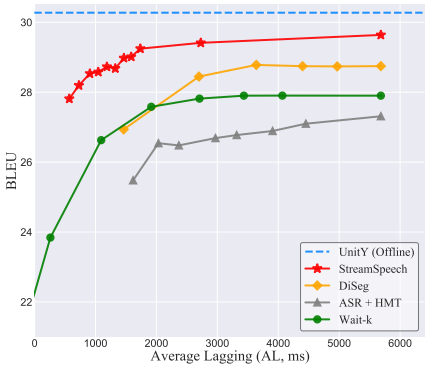



Figure 8: Simultaneous speech-to-text translation results on Fr→En test set.

Totally, the alignments involved in StreamSpeech are reasonable and can be jointly trained with translation through multi-task learning.

### 5.4 Performance on Auxiliary Tasks: Streaming ASR and Simultaneous S2TT

StreamSpeech jointly learns translation and policy through multi-task learning. As an additional product, StreamSpeech can output intermediate results of ASR and S2TT, offering users a more comprehensive experience. To evaluate StreamSpeech's performance on these auxiliary tasks, we present the results on the streaming ASR and Simul-S2TT tasks in Table 5 and Figure 8, respectively, assessed on the CVSS-C Fr→En test set.

For streaming ASR, StreamSpeech achieves performance comparable to Wav2Vec2-large (Baevski et al., 2020) and Whisper-base (Radford et al., 2022) with an average lagging of 100$ms$. For Simul-S2TT, StreamSpeech can generate high-quality translation with an average lagging of 2000$ms$. Overall, StreamSpeech excels in delivering high-quality Simul-S2ST while also providing intermediate results to users as additional references. It's important to note that although intermediate results can be presented, StreamSpeech does not utilize them during inference, but uses hidden states to connect each module, making StreamSpeech a direct Simul-S2ST model.

## 6 Related Work

Recent research often focuses on simultaneous text-to-text (Simul-T2TT) and speech-to-text (Simul-S2TT) translation.

**Simul-T2TT** Simul-T2TT methods fall into fixed and adaptive. For fixed methods, Ma et al. (2019) proposed wait-k policy, which waits for $k$ tokens before alternately READ/WRITE one token (Elbayad et al., 2020; Zheng et al., 2020; Zhang and Feng, 2021a,b). For adaptive methods, monotonic attention (Arivazhagan et al., 2019; Ma et al., 2020c; Zhang and Feng, 2022c,a), alignments (Zhang and Feng, 2023b; Guo et al., 2023a), non-autoregressive architecture (Ma et al., 2023, 2024) or language models (Guo et al., 2024a,c; Zhang et al., 2023) are employed to dynamically perform Simul-T2TT. On top of these policies, some training methods are proposed to enhance the performance of policy (Zhang and Feng, 2022d; Zhang et al., 2022b; Guo et al., 2022, 2023b).

**Simul-S2TT** Simul-S2TT methods focus on the segmentation of speech. Ma et al. (2020b) proposed fixed pre-decision to divide speech into equal-length segments. Some adaptive methods split the speech inputs into words or segments (Ren et al., 2020; Zeng et al., 2021; Chen et al., 2021; Dong et al., 2022; Zhang and Feng, 2022b; Zhang et al., 2022a; Zhang and Feng, 2023a,c), and then apply Simul-T2TT policy. Other methods apply offline model to Simul-S2TT (Papi et al., 2023; Fu et al., 2023; Dugan et al., 2023).

We introduce StreamSpeech, an "All in One" seamless model for offline and simultaneous ASR, translation and synthesis. Compared with SOTA SeamlessStreaming (based on UnitY architecture) (Communication et al., 2023b), StreamSpeech does not design any additional simultaneous policy such as EMMA, but directly jointly learns translation and policy via multi-task learning.

## 7 Conclusion

In this paper, we propose StreamSpeech, an "All in One" seamless model that handles streaming ASR, simultaneous translation and real-time speech synthesis via a unified model. Experiments show the superiority of StreamSpeech on offline S2ST, streaming ASR, simultaneous S2TT and simultaneous S2ST. Moreover, intermediate products such as ASR or S2TT results can also be presented to user during translation process as a reference, offering a better low-latency communication experience.

## Acknowledgements

We thank all the anonymous reviewers for their insightful and valuable comments. This work was supported by a grant from the National Natural Science Foundation of China (No. 62376260).

## Limitations

In this paper, we propose StreamSpeech, a direct Simul-S2ST model that jointly learns translation and policy in a unified framework of multi-task learning. StreamSpeech can achieve high-quality speech-to-speech translation with low latency. However, StreamSpeech currently focuses on synthesizing target speech with a unified voice, which limits its ability to clone the source speech's voice characteristics. Given that voice cloning can enhance the authenticity of low-latency communication, we will explore integrating voice cloning capabilities into StreamSpeech as part of future work.

## 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. pages 1313–1323.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc.

Junkun Chen, Mingbo Ma, Renjie Zheng, and Liang Huang. 2021. Direct simultaneous speech-to-text translation assisted by synchronized streaming ASR. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4618–4624, Online. Association for Computational Linguistics.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation?

Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, Christopher Klaiber, Pengwei Li, Daniel Licht, Jean Maillard, Alice Rakotoarison, Kaushik Ram Sadagopan, Guillaume Wenzek, Ethan Ye, Bapi Akula, Peng-Jen Chen, Naji El Hachem, Brian Ellis, Gabriel Mejia Gonzalez, Justin Haaheim, Prangthip Hansanti, Russ Howes, Bernie Huang, Min-Jae Hwang, Hirofumi Inaguma, Somya Jain, Elahe Kalbassi, Amanda Kallet, Ilia Kulikov, Janice Lam, Daniel Li, Xutai Ma, Ruslan Mavlyutov, Benjamin Peloquin, Mohamed Ramadan, Abinesh Ramakrishnan, Anna Sun, Kevin Tran, Tuan Tran, Igor Tufanov, Vish Vogeti, Carleigh Wood, Yilin Yang, Bokai Yu, Pierre Andrews, Can Balioglu, Marta R. Costa-jussà, Onur Celebi, Maha Elbayad, Cynthia Gao, Francisco Guzmán, Justine Kao, Ann Lee, Alexandre Mourachko, Juan Pino, Sravya Popuri, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, Paden Tomasello, Changhan Wang, Jeff Wang, and Skyler Wang. 2023a. Seamlessm4t: Massively multilingual & multimodal machine translation.

Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Coria Meglioli, David Dale, Ning Dong, Mark Duppenthaler, Paul-Ambroise Duquenne, Brian Ellis, Hady Elsahar, Justin Haaheim, John Hoffman, Min-Jae Hwang, Hirofumi Inaguma, Christopher Klaiber, Ilia Kulikov, Pengwei Li, Daniel Licht, Jean Maillard, Ruslan Mavlyutov, Alice Rakotoarison, Kaushik Ram Sadagopan, Abinesh Ramakrishnan, Tuan Tran, Guillaume Wenzek, Yilin Yang, Ethan Ye, Ivan Evtimov, Pierre Fernandez, Cynthia Gao, Prangthip Hansanti, Elahe Kalbassi, Amanda Kallet, Artyom Kozhevnikov, Gabriel Mejia Gonzalez, Robin San Roman, Christophe Touret, Corinne Wong, Carleigh Wood, Bokai Yu, Pierre Andrews, Can Balioglu, Peng-Jen Chen, Marta R. Costa-jussà, Maha Elbayad, Hongyu Gong, Francisco Guzmán, Kevin Heffernan, Somya Jain, Justine Kao, Ann Lee, Xutai Ma, Alex Mourachko, Benjamin Peloquin, Juan Pino, Sravya Popuri, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, Anna Sun, Paden Tomasello, Changhan Wang, Jeff Wang, Skyler Wang, and Mary Williamson. 2023b. Seamless: Multilingual expressive and streaming speech translation.

Qian Dong, Yaoming Zhu, Mingxuan Wang, and Lei Li. 2022. Learning when to translate for streaming speech. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 680–694, Dublin, Ireland. Association for Computational Linguistics.

Liam Dugan, Anshul Wadhawan, Kyle Spence, Chris Callison-Burch, Morgan McGuire, and Victor Zordan. 2023. Learning when to speak: Latency and quality trade-offs for simultaneous speech-to-speech translation with offline models.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2020. Efficient Wait-k Models for Simultaneous Machine Translation.

Qingkai Fang, Yan Zhou, and Yang Feng. 2023. Daspeech: Directed acyclic transformer for fast and high-quality speech-to-speech translation.

Biao Fu, Minpeng Liao, Kai Fan, Zhongqiang Huang, Boxing Chen, Yidong Chen, and Xiaodong Shi. 2023. Adapting offline speech translation models for streaming with future-aware distillation and inference. In *Proceedings of the 2023 Conference on*

*Empirical Methods in Natural Language Processing*, pages 16600–16619, Singapore. Association for Computational Linguistics.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine translation*, 21:209–252.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA. Association for Computing Machinery.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proc. Interspeech 2020*, pages 5036–5040.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2022. Turning fixed to adaptive: Integrating post-evaluation into simultaneous machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2264–2278, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2023a. Learning optimal policy for simultaneous machine translation via binary search. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2318–2333, Toronto, Canada. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2023b. Simultaneous machine translation with tailored reference. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3070–3084, Singapore. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2024a. Decoder-only streaming transformer for simultaneous translation. In *Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2024b. Glancing future for simultaneous machine translation. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11386–11390.

Shoutao Guo, Shaolei Zhang, Zhengrui Ma, Min Zhang, and Yang Feng. 2024c. Sillm: Large language models for simultaneous machine translation.

Hirofumi Inaguma, Sravya Popuri, Ilia Kulikov, Peng-Jen Chen, Changhan Wang, Yu-An Chung, Yun Tang, Ann Lee, Shinji Watanabe, and Juan Pino. 2023. UnitY: Two-pass direct speech-to-speech translation with discrete units. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15655–15680, Toronto, Canada. Association for Computational Linguistics.

Ye Jia, Michelle Tadmor Ramanovich, Tal Remez, and Roi Pomerantz. 2022a. Translatotron 2: High-quality direct speech-to-speech translation with voice preservation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10120–10134. PMLR.

Ye Jia, Michelle Tadmor Ramanovich, Quan Wang, and Heiga Zen. 2022b. CVSS corpus and massively multilingual speech-to-speech translation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6691–6703, Marseille, France. European Language Resources Association.

Ye Jia, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen, and Yonghui Wu. 2019. Direct Speech-to-Speech Translation with a Sequence-to-Sequence Model. In *Proc. Interspeech 2019*, pages 1123–1127.

Yasumasa Kano, Katsuhito Sudoh, and Satoshi Nakamura. 2023. Average token delay: A latency metric for simultaneous translation.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Advances in Neural Information Processing Systems*, volume 33, pages 17022–17033. Curran Associates, Inc.

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*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Ann Lee, Peng-Jen Chen, Changhan Wang, Jiatao Gu, Sravya Popuri, Xutai Ma, Adam Polyak, Yossi Adi, Qing He, Yun Tang, Juan Pino, and Wei-Ning Hsu. 2022. Direct speech-to-speech translation with discrete units. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*

*(Volume 1: Long Papers)*, pages 3327–3339, Dublin, Ireland. 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, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020a. SIMULEVAL: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150, Online. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020b. 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*, pages 582–587, Suzhou, China. Association for Computational Linguistics.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020c. Monotonic multihead attention. In *International Conference on Learning Representations*.

Zhengrui Ma, Qingkai Fang, Shaolei Zhang, Shoutao Guo, Yang Feng, and Min Zhang. 2024. A non-autoregressive generation framework for end-to-end simultaneous speech-to-any translation. In *Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Zhengrui Ma, Shaolei Zhang, Shoutao Guo, Chenze Shao, Min Zhang, and Yang Feng. 2023. Non-autoregressive streaming transformer for simultaneous translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5177–5190, Singapore. Association for Computational Linguistics.

OpenAI. 2024. Hello gpt-4o.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022. Over-generation cannot be rewarded: Length-adaptive average lagging for simultaneous speech translation. In *Proceedings of the Third Workshop on Automatic Simultaneous Translation*, pages 12–17, Online. Association for Computational Linguistics.

Sara Papi, Matteo Negri, and Marco Turchi. 2023. Attention as a guide for simultaneous speech translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13340–13356, Toronto, Canada. Association for Computational Linguistics.

Sravya Popuri, Peng-Jen Chen, Changhan Wang, Juan Pino, Yossi Adi, Jiatao Gu, Wei-Ning Hsu, and Ann Lee. 2022. Enhanced Direct Speech-to-Speech Translation Using Self-supervised Pre-training and Data Augmentation. In *Proc. Interspeech 2022*, pages 5195–5199.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision.

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*, pages 3787–3796, Online. Association for Computational Linguistics.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online. Association for Computational Linguistics.

Elizabeth Salesky, Marcello Federico, and Marine Carpuat, editors. 2023. *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*. Association for Computational Linguistics, Toronto, Canada (in-person and online).

Yun Tang, Juan Pino, Xian Li, Changhan Wang, and Dmitriy Genzel. 2021a. Improving speech translation by understanding and learning from the auxiliary text translation task. In *Proceedings of the 59th Annual Meeting of the Association for Computational*

*Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4252–4261, Online. Association for Computational Linguistics.

Yun Tang, Juan Pino, Changhan Wang, Xutai Ma, and Dmitriy Genzel. 2021b. A general multi-task learning framework to leverage text data for speech to text tasks. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6209–6213.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Changhan Wang, Anne Wu, and Juan Pino. 2020. Covost 2: A massively multilingual speech-to-text translation corpus.

Xingshan Zeng, Liangyou Li, and Qun Liu. 2021. RealTranS: End-to-end simultaneous speech translation with convolutional weighted-shrinking transformer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2461–2474, Online. Association for Computational Linguistics.

Ruiqing Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2022a. Learning adaptive segmentation policy for end-to-end simultaneous translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7862–7874, Dublin, Ireland. Association for Computational Linguistics.

Shaolei Zhang, Qingkai Fang, Zhuocheng Zhang, Zhengrui Ma, Yan Zhou, Langlin Huang, Mengyu Bu, Shangtong Gui, Yunji Chen, Xilin Chen, and Yang Feng. 2023. Bayling: Bridging cross-lingual alignment and instruction following through interactive translation for large language models.

Shaolei Zhang and Yang Feng. 2021a. ICT's system for AutoSimTrans 2021: Robust char-level simultaneous translation. In *Proceedings of the Second Workshop on Automatic Simultaneous Translation*, pages 1–11, Online. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2021b. Universal simultaneous machine translation with mixture-of-experts wait-k policy. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7306–7317, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022a. Gaussian multi-head attention for simultaneous machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3019–3030, Dublin, Ireland. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022b. Information-transport-based policy for simultaneous translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 992–1013, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022c. Modeling dual read/write paths for simultaneous machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2461–2477, Dublin, Ireland. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022d. Reducing position bias in simultaneous machine translation with length-aware framework. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6775–6788, Dublin, Ireland. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2023a. End-to-end simultaneous speech translation with differentiable segmentation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7659–7680, Toronto, Canada. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2023b. Hidden markov transformer for simultaneous machine translation. In *The Eleventh International Conference on Learning Representations*.

Shaolei Zhang and Yang Feng. 2023c. Unified segment-to-segment framework for simultaneous sequence generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 45235–45258. Curran Associates, Inc.

Shaolei Zhang, Yang Feng, and Liangyou Li. 2021. Future-guided incremental transformer for simultaneous translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14428–14436.

Shaolei Zhang, Shoutao Guo, and Yang Feng. 2022b. Wait-info policy: Balancing source and target at information level for simultaneous machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2249–2263, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2847–2853, Online. Association for Computational Linguistics.

## A Calculation of Expected Token Number in CTC sequence

StreamSpeech leverages ASR and NAR-S2TT tasks to learn the alignment between the source speech and the source/target text, and then makes READ/WRITE decisions based on the token number corresponding to the current received speech. Since the alignments are obtained through the CTC decoder, which involves blank and repeated tokens, we need to count the number of tokens that can be decoded by the CTC sequence. During inference, it is straightforward to remove duplicates and blanks from the CTC sequence to count the tokens corresponding to the speech. However, during training, we calculate the expected number of tokens corresponding to the CTC sequence.

For the excepted number of source tokens aligned to the speech inputs $X$, $\mathcal{N}_j^{asr}$ is calculated as:

$$\mathcal{N}_j^{asr} = \sum_{m=1}^{j} \Bigg( 1 - p\left(\phi \mid D_m^{asr}\right) - \sum_{v \in \mathcal{V}} \Big( p\left(v \mid D_m^{asr}\right) \times p\left(v \mid D_{m-1}^{asr}\right) \Big) \Bigg). \tag{13}$$

where $\mathcal{N}_j^{asr}$ is the number of source tokens that align to $X_{\leq j}$. $\mathcal{N}_j^{asr}$ are calculated in an expectation manner, where $p\left(\phi | D_m^{asr}\right)$ refers to generating blank token and $\sum_{v \in \mathcal{V}} (p\left(v | D_m^{asr}\right) \times p\left(v | D_{m-1}^{asr}\right))$ refers to generating repetitive tokens over the vocabulary $\mathcal{V}$. Similarly, the number of target tokens that align to $X_{\leq j}$ is calculated in the same way, denoted as $\mathcal{N}_j^{nar\text{-}s2tt}$. In particular, the probabilities within the CTC sequence often tend to be discretized, resulting in minimal differences between the token counts in training and inference.

## B Details of Ablation Study on Policy

In Sec.5.3, we conduct an ablation study on the three alignments involved in StreamSpeech, including source speech and source text, source speech and target text, target text and target speech. Here, we provide detailed explanations of the ablation study settings.

When removing the alignment between the source speech and source text, StreamSpeech is no longer wait for recognition of new tokens corresponding to the source speech but directly controls READ/WRITE based on the number of target tokens corresponding to the source speech. When removing the alignment between the source speech and target text, StreamSpeech generates one target token once recognizing one source token. When removing the alignment between target text and target speech captured by the NAR T2U module, the second pass of StreamSpeech adopts the same autoregressive architecture as UnitY and generates units autoregressively until <eos>.

| StreamSpeech | $r = 15$ | $r = 20$ | $r = 25$ | $r = 30$ |
|---|---|---|---|---|
| **BLEU of units** | 30.48 | 31.08 | **32.00** | 31.55 |
| **ASR-BLEU** | 26.72 | 26.75 | **28.48** | 27.91 |

Table 6: Performance of various upsampling rate $r$ on CVSS-C Fr→En validation set. "BLEU of units": BLEU score computed on generated units sequence.

## C Detailed Introduction of Baselines

Here, we give a detailed introduction to offline S2ST baselines.

**S2UT** (Lee et al., 2022) Speech-to-unit translation (S2UT) directly translates the source speech to the target unit via one-pass architecture.

**Translatotron** (Jia et al., 2019) Translatotron translates the source speech to the target mel-spectrogram via one-pass architecture.

**Translatotron 2** (Jia et al., 2022a) Translatotron 2 employs a two-pass architecture that generates phonemes and mel-spectrogram successively.

**DASpeech** (Fang et al., 2023) DASpeech employs a two-pass architecture, where first performs non-autoregressive translation and then generates mel-spectrogram via fastspeech 2.

**UnitY** (Inaguma et al., 2023) UnitY is the state-of-the-art S2ST model, where both first and second passes apply autoregressive encoder-decoder to generate target text and units successively.

For the cascaded Simul-S2ST system, we employ state-of-the-art methods, HMT from Simul-T2TT and DiSeg from Simul-S2TT, in conjunction with streaming ASR and real-time TTS module to accomplish Simul-S2ST.

**ASR+HMT+TTS** (Zhang and Feng, 2023b) Hidden Markov Transformer (HMT), which uses a hidden Markov model to correspond source tokens with the target tokens, thereby learning the optimal translating moments for generating each target token.

**DiSeg+TTS** (Zhang and Feng, 2023a) DiSeg learns the speech segmentation from the underlying translation model via the differentiable segmen-

| Models | Fr→En | | Es→En | | De→En | | Average | |
|---|---|---|---|---|---|---|---|---|
| | greedy | beam10 | greedy | beam10 | greedy | beam10 | greedy | beam10 |
| *ASR-BLEU* | | | | | | | | |
| **UnitY** | 26.90 | 27.77 | 23.93 | 24.95 | 18.19 | 18.74 | 23.01 | 23.82 |
| **StreamSpeech** | **27.58** | **28.45** | **26.16** | **27.25** | **19.72** | **20.93** | **24.49** | **25.54** |
| *BLASER 2.0 (Unsupervised)* | | | | | | | | |
| **UnitY** | 0.4467 | 0.4473 | 0.5090 | 0.5116 | 0.4431 | 0.4435 | 0.4663 | 0.4674 |
| **StreamSpeech** | **0.4486** | **0.4491** | **0.5155** | **0.5178** | **0.4514** | **0.4544** | **0.4719** | **0.4738** |
| *BLASER 2.0 (QE)* | | | | | | | | |
| **UnitY** | 3.1674 | 3.1772 | 3.3020 | 3.3278 | 3.1322 | 3.1537 | 3.2006 | 3.2195 |
| **StreamSpeech** | **3.1779** | **3.1872** | **3.3442** | **3.3669** | **3.1698** | **3.2033** | **3.2307** | **3.2525** |
| *BLASER 2.0 (Ref)* | | | | | | | | |
| **UnitY** | 3.1744 | 3.1965 | 3.2213 | 3.2638 | 2.9125 | 2.9372 | 3.1028 | 3.1325 |
| **StreamSpeech** | **3.1989** | **3.2200** | **3.3146** | **3.3525** | **3.0008** | **3.0482** | **3.1714** | **3.2069** |

Table 7: Offline S2ST performance of StreamSpeech, evaluated with BLASER 2.0.

tation, and then apply wait-k policy based on the number of speech segments.

## D Upsampling Rate in NAR T2U Generation

The only hyperparameter that needs to be set in StreamSpeech is the upsampling rate $r$ in NAR T2U generation. Table 6 reports the offline S2ST performance with different $r$ on the CVSS-C Fr→En validation set, with $r = 25$ achieving the best performance. This finding is consistent with previous conclusions in non-autoregressive translation (NAT), where an upsampling rate of 2-3 times yielded the best performance (Saharia et al., 2020). Therefore, we set $r = 25$ in our experiments accordingly. The unit sequence length is approximately 10 times that of the subword sequence length, and with a 2-3 times upsampling rate, an overall upsampling rate of around 25 times from text sequence to unit sequence is optimal.

When training StreamSpeech for a new language, it is recommended to first estimate the length ratio between the unit sequence and the subword sequence, and then multiply this ratio by 2-3 times to determine the appropriate upsampling rate.

## E Evaluation with BLASER 2.0

Besides ASR-BLEU, we use BLASER 2.0[9] to assess the quality of the generated speech. BLASER 2.0 leverages a multilingual multimodal encoder to directly encode the speech segments for source input, translation output, and reference into a shared embedding space. It then computes a score of the translation quality that can serve as a proxy for human evaluation. BLASER 2.0 comprises three versions: Unsupervised (score 0-1), QE (score 1-5), and Ref (score 1-5). Table 7 reports the offline S2ST performance of StreamSpeech evaluated by BLASER 2.0. StreamSpeech also has significant advantages over UnitY.

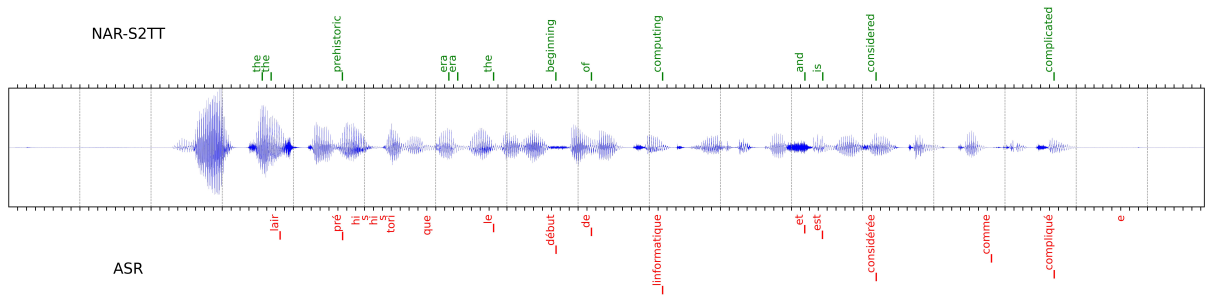
## F Visualization of Alignments

The policy of StreamSpeech is primarily guided by the alignment between source speech and source/target text, which is captured through the CTC decoder of the introduced ASR and NAR-S2TT tasks. We visualize the alignment captured by the CTC decoder in Figure 9.

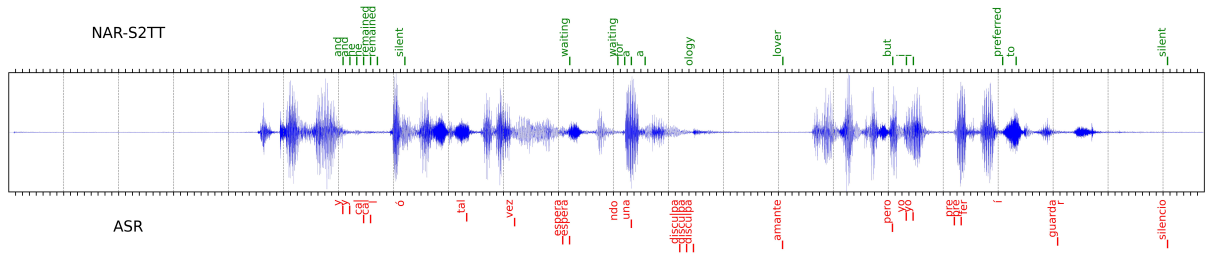
The CTC decoder of the ASR and NAR-S2TT tasks effectively captures the alignment of speech and text and generates tokens with high accuracy, especially in terms of 1-gram accuracy. StreamSpeech starts translating upon recognizing a new source token and generates a corresponding number of target words. This ensures that the received speech before translation contains complete source tokens and provides sufficient information to generate target tokens.

Additionally, we observe that certain tokens occupying the same position in the ASR and NAR-S2TT CTC sequences correspond to the same semantic meaning. For example, '*début*'↔'*beginning*', '*linformatique*'↔'*computing*' in Fr→En, '*amante*'↔'*lover*', '*silencio*'↔'*silent*' in Es→En, '*Dort*'↔'*there*', '*back*'↔'*stream*' in De→En. This suggests that

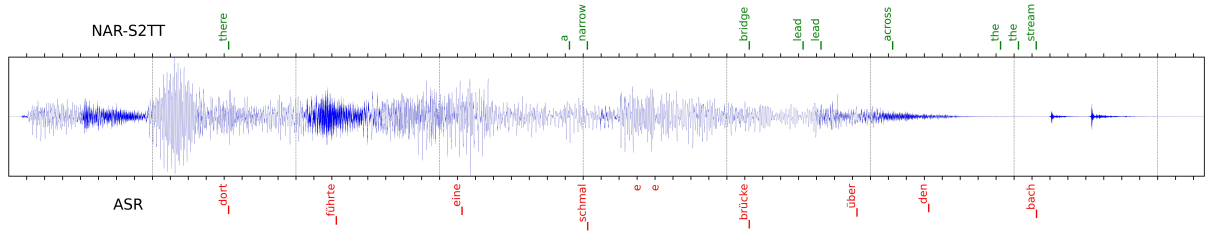
[9] https://facebookresearch.github.io/stopes/docs/eval/blaser




(a) Case common_voice_fr_17308913 in CVSS-C Fr→En. Source transcription: *laire préhistorique est le début de linformatique et est considéré comme compliquer*. Target translation: *the prehistoric area is the beginning of computer science and is considered to be complicated*.




(b) Case common_voice_es_18307761 in CVSS-C Es→En. Source transcription: *y calló tal vez esperando una disculpa amante pero yo preferí guardar silencio*. Target translation: *and he shut up he might have been just waiting for a loving apology but i preferred to remain silent*.



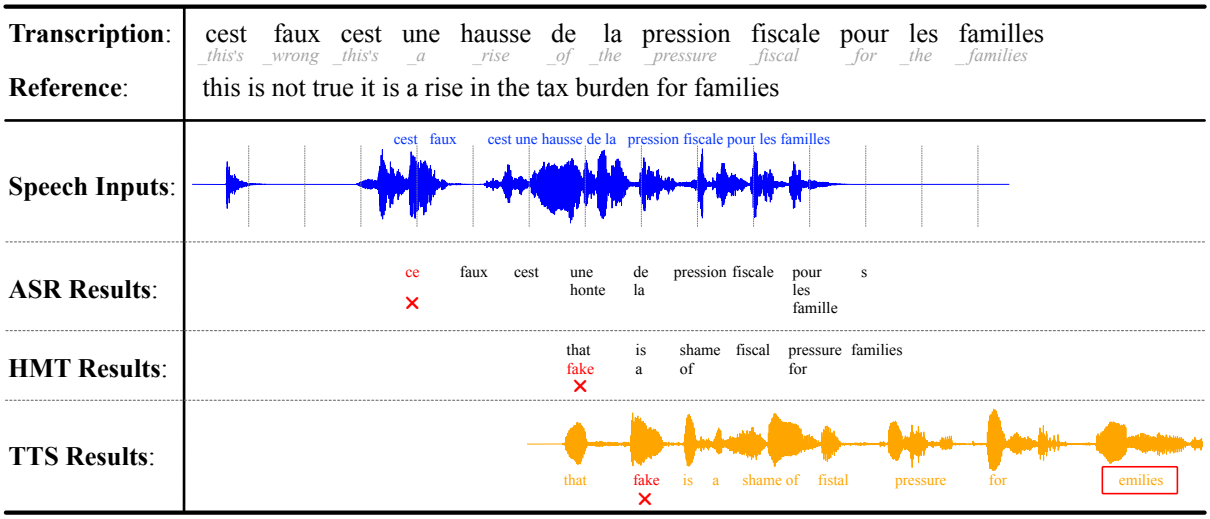

(c) Case common_voice_de_17300640 in CVSS-C De→En. Source transcription: *dort führt eine schmale brücke über den bach*. Target translation: *there a narrow bridge leads over the stream.*

Figure 9: Visualization of the alignments of source speech and source/target text within the CTC decoder for ASR and NAR-S2TT tasks. Note that the positions without label refer to generating blank token $\phi$, and we omit them for clarity. The vertical grey dashed lines represent chunks of $320ms$.

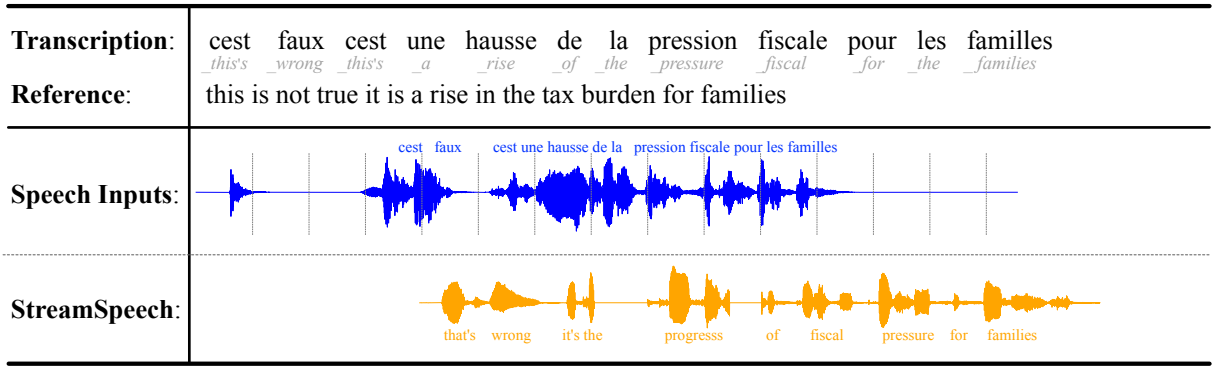
the introduction of both source and target language CTC decoders after the encoder implicitly models cross-lingual alignments, particularly given that our introduced CTC decoder consists solely of a single fully connected layer.

## G Case Study

In Figure 10, we illustrate the Simul-S2ST process of direct StreamSpeech and the cascaded "ASR+HMT+TTS" system. StreamSpeech is capable of generating high-quality target speech with a delay of 2 seconds, particularly noticeable when there is prolonged silence at the beginning of the source speech. Compared to the cascaded system, direct StreamSpeech also demonstrates clear advantages. Specifically, the cascaded system requires streaming ASR to first transcribe the speech into source text, then translate the current source text into target text using state-of-the-art HMT, and finally synthesize the target speech. The cascading of multiple modules leads to error accumulation, especially as the accuracy of each module in streaming scenarios generally tends to be lower than in offline scenarios. For instance, in this case, streaming ASR may incorrectly transcribe '*cest*' as '*ce*', leading to subsequent HMT generating the erroneous '*fake*', ultimately resulting in incorrect speech. Therefore, for more challenging simultaneous scenarios, direct models hold an advantage over cascaded systems.




(a) Streaming ASR + HMT + Real-time TTS




(b) StreamSpeech

Figure 10: Case study of direct StreamSpeech and cascaded 'ASR+HMT+TTS'. For clarity, we have marked the blue text above the source audio to represent the ground truth transcription aligned with the speech. The orange text below the target speech indicates the text transcribed by ASR-BLEU tookit.

You can hear more cases of StreamSpeech on offline or simultaneous speech-to-speech translation at our project page (https://ictnlp.github.io/StreamSpeech-site/).

## H Configuration and Training Details

Table 8 reports the configurations of StreamSpeech and baselines. For the offline scenario, we set the chunk size of StreamSpeech to infinity and do not involve simultaneous policy, while keeping all other settings identical to the simultaneous scenario. For the simultaneous scenario, to evaluate Simul-S2ST under different latencies, we employ multi-chunk training to train a single StreamSpeech model and utilize this model to perform Simul-S2ST under various latency. All models are trained on 4 NVIDIA RTX 3090 GPUs.

## I Latency Metrics

To more comprehensively evaluate the latency of simultaneous speech-to-speech translation, we employ a variety of latency metrics, which are mainly divided into three categories: latency, computation-aware latency and streaming degree. To compute the latency metrics, we record the moment when the $i^{th}$ frame of the target speech is generated as $t_i$ (the starting point of the source speech considered as moment 0), and $X$ and $S$ are the source speech and target speech, respectively. Note that all metrics are automatically calculated via SimulEval toolkit.

Latency evaluates the duration that outputs lag behind the inputs, including:

**Average Lagging (AL)** (Ma et al., 2019) evaluates the average speech duration that target outputs

lag behind the source inputs. AL is calculated as:

$$\mathrm{AL} = \frac{1}{\tau}\sum_{i=1}^{\tau} t_i - \frac{i-1}{|S|/|X|}, \quad \text{where } \tau = \underset{i}{\operatorname{argmin}}\left(t_i = |X|\right). \tag{14}$$

**Average Proportion (AP)** (Cho and Esipova, 2016) evaluates the proportion between the generating moment and the total duration of source speech, calculated as:

$$\mathrm{AP} = \frac{1}{|X|\,|S|}\sum_{i=1}^{|S|} t_i. \tag{15}$$

**Differentiable Average Lagging (DAL)** (Arivazhagan et al., 2019) is a differentiable version of average lagging, calculated as:

$$\mathrm{DAL} = \frac{1}{|S|}\sum_{i=1}^{|S|} t_i^{'} - \frac{i-1}{|S|/|X|}, \quad \text{where } t_i^{'} = \begin{cases} t_i & i = 1 \\ \max\left(t_i, t_{i-1}^{'} + \frac{|X|}{|S|}\right) & i > 1 \end{cases}. \tag{16}$$

**StartOffset** measures the waiting time before outputting the first frame of target speech, calculated as:

$$\mathrm{StartOffset} = t_1 \tag{17}$$

**EndOffset** measures the offset of the last frame of target speech relative to the end of source speech, calculated as:

$$\mathrm{EndOffset} = t_{|S|} - |X| \tag{18}$$

**Length-Adaptive Average Lagging (LAAL)** (Papi et al., 2022) is a modified version of the average lagging that takes into account the over-generation phenomenon, calculated as:

$$\mathrm{LAAL} = \frac{1}{\tau}\sum_{i=1}^{\tau} t_i - \frac{i-1}{\max(|S|, |S^*|)/|X|}, \quad \text{where } \tau = \underset{i}{\operatorname{argmin}}\left(t_i = |X|\right), \tag{19}$$

where $S^*$ is generated target speech.

**Average Token Delay (ATD)** (Kano et al., 2023) is the average delay of output sub-segments against their corresponding input sub-segments, calculated as:

$$\mathrm{ATD} = \frac{1}{|S|}\sum_{i=1}^{|S|} t_i - \xi_{seg_{t_i}}, \tag{20}$$

where $\xi_{seg_{t_i}}$ is the moment of corresponding input sub-segments of the $i^{th}$ output.

Computation-aware latency considers the actual inference time of the model when computing the aforementioned latency, including: AL_CA, AP_CA, DAL_CA, StartOffset_CA, EndOffset_CA, LAAL_CA and ATD_CA.

Besides, we also evaluate the streaming degree of the generated speech. The more segments of output speech generated with shorter durations for each segment, the closer the generation is to being considered streaming. The metrics include:

**Number of Chunks (NumChunks)** evaluates the number of segments when generating the target speech.

**Discontinuity** evaluates the duration of silence produced in the generated speech while waiting for the source speech. This includes the total duration of all silences (Sum), the average duration of each silence (Ave), and the number of silence segments (Num). It's important to note that Discontinuity is not equivalent to NumChunks. When the model finishes generating a target speech segment (i.e., a chunk), if the incoming source speech is sufficient for the model to begin translation at that moment immediately, the model will not produce discontinuity.

**Real-time Factor (RTF)** (Fügen et al., 2007) describes the ratio between the duration of outputs and inputs.

For more detailed implementations of latency computation, please refer to SimulEval toolkit[10].

## J Numerical Results

Tables 9, 10 and 11 report the numerical results of StreamSpeech, including more comprehensive quality and latency metrics.

**ASR-BLEU (with silence)** In particular, we additionally calculate the ASR-BLEU considering silence for quality evaluation, denoted as *ASR-BLEU (with silence)*. Specifically, StreamSpeech remains silent while waiting for the source speech after generating the current speech outputs. For

[10] https://github.com/facebookresearch/SimulEval/blob/main/simuleval/evaluator/scorers/latency_scorer.py

instance, if StreamSpeech generates the current speech of 220$ms$, it will continue to wait for the streaming speech inputs of 320ms (corresponding to the chunk size). During the 100ms interval between 220ms and the next 320ms chunk, StreamSpeech remains silent. ASR-BLEU (with silence) is calculated directly on these speech outputs that include silence. Note that the ASR model used in ASR-BLEU was trained on standard continuous speech, which causes a mismatch when evaluating speech with silence (may lead to some recognition errors). Nevertheless, we report this metric to provide a more comprehensive evaluation of StreamSpeech.

| Hyperparameters | S2UT | Translatotron | Translatotron2 | DASpeech | UnitY | StreamSpeech |
|---|---|---|---|---|---|---|
| *Speech Encoder* | | | | | | |
| conv_kernel_sizes | (5, 5) | (5, 5) | (5, 5) | (5, 5) | (5, 5) | (5, 5) |
| encoder_type | Conformer | Conformer | Conformer | Conformer | Conformer | Conformer |
| encoder_layers | 12 | 12 | 12 | 12 | 12 | 12 |
| encoder_embed_dim | 256 | 256 | 256 | 256 | 256 | 256 |
| encoder_ffn_embed_dim | 2048 | 2048 | 2048 | 2048 | 2048 | 2048 |
| encoder_attention_heads | 4 | 4 | 4 | 4 | 4 | 4 |
| encoder_pos_enc_type | relative | relative | relative | relative | relative | relative |
| depthwise_conv_kernel_size | 31 | 31 | 31 | 31 | 31 | 31 |
| streaming | × | × | × | × | × | ✓ |
| *Text decoder* | | | | | | |
| decoder_type | Transformer | Transformer | Transformer | Transformer | Transformer | Transformer |
| decoder_layers | 4 | 4 | 4 | 4 | 4 | 4 |
| decoder_embed_dim | 512 | 512 | 512 | 512 | 512 | 512 |
| decoder_ffn_embed_dim | 2048 | 2048 | 2048 | 2048 | 2048 | 2048 |
| decoder_attention_heads | 8 | 8 | 8 | 8 | 8 | 8 |
| *Text-to-Speech Encoder* | | | | | | |
| encoder_type | - | - | Transformer | - | Transformer | Transformer |
| encoder_layers | - | - | 2 | - | 2 | 2 |
| encoder_embed_dim | - | - | 512 | - | 512 | 512 |
| encoder_ffn_embed_dim | - | - | 2048 | - | 2048 | 2048 |
| encoder_attention_heads | - | - | 8 | - | 8 | 8 |
| *Acoustic Decoder* | | | | | | |
| output_type | unit (1000) | mel-spectrogram | mel-spectrogram | mel-spectrogram | unit (1000) | unit (1000) |
| decoder_layers | 6 | 6 | 6 | 6 | 2 | 2 |
| decoder_embed_dim | 512 | 512 | 512 | 512 | 512 | 512 |
| decoder_ffn_embed_dim | 2048 | 2048 | 2048 | 2048 | 2048 | 2048 |
| decoder_attention_heads | 8 | 8 | 8 | 4 | 8 | 8 |
| *Training* | | | | | | |
| lr | 1e-3 | 1e-3 | 1e-3 | 1e-3 | 1e-3 | 1e-3 |
| lr_scheduler | inverse_sqrt | inverse_sqrt | inverse_sqrt | inverse_sqrt | inverse_sqrt | inverse_sqrt |
| warmup_updates | 4000 | 4000 | 4000 | 4000 | 4000 | 4000 |
| warmup_init_lr | 1e-7 | 1e-7 | 1e-7 | 1e-7 | 1e-7 | 1e-7 |
| optimizer | Adam | Adam | Adam | Adam | Adam | Adam |
| dropout | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| weight_decay | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| clip_norm | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| max_tokens | 160k | 160k | 160k | 160k | 160k | 160k |
| s2st_loss_weight | 1.0 | 1.0 | 1.0 | 5.0 | 1.0 | 1.0 |
| s2tt_loss_weight | 8.0 | 0.1 | 0.1 | 1.0 | 8.0 | 8.0 |
| nar_s2tt_loss_weight | - | - | - | - | - | 4.0 |
| asr_loss_weight | - | - | - | - | - | 4.0 |

Table 8: Configuration of StreamSpeech and baselines.

| CVSS-C Fr→En | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $C$×40$ms$ | **ASR-BLEU** | **Latency** | | | | | | |
| | | AL | AP | DAL | StartOffset | EndOffset | LAAL | ATD |
| 320 $ms$ | 22.89 | 1269.84 | 0.52 | 1702.30 | 1667.11 | 699.22 | 1358.15 | 2290.69 |
| 640 $ms$ | 24.41 | 2326.17 | 0.40 | 1946.50 | 1888.58 | 1030.48 | 2332.34 | 2669.34 |
| 960 $ms$ | 25.00 | 2803.13 | 0.35 | 2124.49 | 2076.37 | 1107.48 | 2806.27 | 2862.42 |
| 1280 $ms$ | 25.20 | 3146.27 | 0.31 | 2309.87 | 2231.73 | 1211.81 | 3148.17 | 3079.90 |
| 1600 $ms$ | 25.30 | 3287.21 | 0.29 | 2352.54 | 2215.25 | 1321.50 | 3288.82 | 3240.42 |
| 1920 $ms$ | 25.50 | 3450.24 | 0.27 | 2477.95 | 2296.93 | 1436.70 | 3451.45 | 3381.28 |
| 2240 $ms$ | 25.50 | 3629.71 | 0.25 | 2666.28 | 2471.04 | 1545.23 | 3630.60 | 3520.22 |
| 2560 $ms$ | 25.68 | 3812.13 | 0.24 | 2891.02 | 2695.36 | 1639.45 | 3812.69 | 3651.32 |
| 2880 $ms$ | 25.60 | 3992.35 | 0.22 | 3131.35 | 2957.32 | 1719.35 | 3992.80 | 3776.42 |
| 3200 $ms$ | 25.75 | 4157.28 | 0.22 | 3370.39 | 3228.57 | 1800.14 | 4157.49 | 3908.66 |
| 4800 $ms$ | 26.14 | 4873.08 | 0.18 | 4505.76 | 4490.42 | 2250.83 | 4873.08 | 4640.12 |
| 10000 $ms$ | 26.20 | 5683.92 | 0.13 | 5683.92 | 5683.92 | 3096.54 | 5683.92 | 5672.35 |
| $C$×40$ms$ | **ASR-BLEU** | **Computation-Aware Latency** | | | | | | |
| | | AL_CA | AP_CA | DAL_CA | StartOffset_CA | EndOffset_CA | LAAL_CA | ATD_CA |
| 320 $ms$ | 22.89 | 2195.04 | 0.68 | 2333.37 | 2052.33 | 699.22 | 2269.02 | 2840.15 |
| 640 $ms$ | 24.41 | 2908.52 | 0.47 | 2369.64 | 2164.57 | 1030.48 | 2913.75 | 2958.70 |
| 960 $ms$ | 25.00 | 3331.47 | 0.41 | 2548.80 | 2363.50 | 1107.48 | 3334.27 | 3133.06 |
| 1280 $ms$ | 25.20 | 3576.95 | 0.35 | 2680.58 | 2469.84 | 1211.81 | 3578.67 | 3292.64 |
| 1600 $ms$ | 25.30 | 3694.33 | 0.33 | 2756.06 | 2451.59 | 1321.50 | 3695.72 | 3449.34 |
| 1920 $ms$ | 25.50 | 3765.89 | 0.29 | 2777.31 | 2465.22 | 1436.70 | 3766.86 | 3540.84 |
| 2240 $ms$ | 25.50 | 3995.11 | 0.28 | 3029.87 | 2677.67 | 1545.23 | 3995.87 | 3725.77 |
| 2560 $ms$ | 25.68 | 4167.48 | 0.26 | 3229.14 | 2920.50 | 1639.45 | 4168.00 | 3837.85 |
| 2880 $ms$ | 25.60 | 4323.54 | 0.24 | 3430.48 | 3170.37 | 1719.35 | 4323.93 | 3951.69 |
| 3200 $ms$ | 25.75 | 4502.11 | 0.23 | 3671.07 | 3406.31 | 1800.14 | 4502.32 | 4105.75 |
| 4800 $ms$ | 26.14 | 5189.20 | 0.19 | 4752.24 | 4723.36 | 2250.83 | 5189.20 | 4815.85 |
| 10000 $ms$ | 26.20 | 5946.97 | 0.13 | 5946.97 | 5946.97 | 3096.54 | 5946.97 | 5816.81 |
| $C$×40$ms$ | **ASR-BLEU** | **Streaming Degree** | | | | | | |
| | | Num Chunks | RTF | Discontinuity Sum | Discontinuity Ave | Discontinuity Num | | |
| 320 $ms$ | 22.89 | 7.85 | 1.15 | 1695.99 | 385.77 | 4.42 | | |
| 640 $ms$ | 24.41 | 5.43 | 1.20 | 1745.35 | 549.87 | 3.28 | | |
| 960 $ms$ | 25.00 | 4.46 | 1.22 | 1630.69 | 585.05 | 2.72 | | |
| 1280 $ms$ | 25.20 | 3.83 | 1.24 | 1583.53 | 672.25 | 2.23 | | |
| 1600 $ms$ | 25.30 | 3.48 | 1.26 | 1705.61 | 843.01 | 1.93 | | |
| 1920 $ms$ | 25.50 | 3.15 | 1.29 | 1736.51 | 997.20 | 1.64 | | |
| 2240 $ms$ | 25.50 | 2.86 | 1.30 | 1668.21 | 1101.78 | 1.37 | | |
| 2560 $ms$ | 25.68 | 2.62 | 1.32 | 1543.38 | 1144.47 | 1.16 | | |
| 2880 $ms$ | 25.60 | 2.40 | 1.34 | 1361.69 | 1103.53 | 0.97 | | |
| 3200 $ms$ | 25.75 | 2.23 | 1.35 | 1166.73 | 1009.29 | 0.80 | | |
| 4800 $ms$ | 26.14 | 1.69 | 1.44 | 356.97 | 354.08 | 0.25 | | |
| 10000 $ms$ | 26.20 | 1.00 | 1.56 | 0.00 | 0.00 | 0.00 | | |
| $C$×40$ms$ | **ASR-BLEU** | **Quality with other Metrics** | | | | | | |
| | | ASR-BLEU (with silence) | | BLASER 2.0 (Unsupervised) | BLASER 2.0 (QE) | BLASER 2.0 (Ref) | | |
| 320 $ms$ | 22.89 | 17.88 | | 0.4428 | 3.1170 | 3.0519 | | |
| 640 $ms$ | 24.41 | 19.65 | | 0.4439 | 3.1322 | 3.0965 | | |
| 960 $ms$ | 25.00 | 20.20 | | 0.4446 | 3.1373 | 3.1067 | | |
| 1280 $ms$ | 25.20 | 21.14 | | 0.4448 | 3.1391 | 3.1109 | | |
| 1600 $ms$ | 25.30 | 21.07 | | 0.4450 | 3.1420 | 3.1164 | | |
| 1920 $ms$ | 25.50 | 21.61 | | 0.4451 | 3.1429 | 3.1195 | | |
| 2240 $ms$ | 25.50 | 21.97 | | 0.4451 | 3.1443 | 3.1226 | | |
| 2560 $ms$ | 25.68 | 22.76 | | 0.4456 | 3.1464 | 3.1268 | | |
| 2880 $ms$ | 25.60 | 23.26 | | 0.4456 | 3.1484 | 3.1294 | | |
| 3200 $ms$ | 25.75 | 23.91 | | 0.4456 | 3.1475 | 3.1294 | | |
| 4800 $ms$ | 26.14 | 25.66 | | 0.4462 | 3.1530 | 3.1397 | | |
| 10000 $ms$ | 26.20 | 26.20 | | 0.4465 | 3.1569 | 3.1486 | | |

Table 9: Numerical results of StreamSpeech on CVSS-C Fr→En.

| CVSS-C Es→En | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $C$×40$ms$ | **ASR-BLEU** | **Latency** | | | | | | |
| | | AL | AP | DAL | StartOffset | EndOffset | LAAL | ATD |
| 320 $ms$ | 20.06 | 1522.05 | 0.52 | 1899.15 | 1829.94 | 811.60 | 1611.94 | 2647.52 |
| 640 $ms$ | 21.68 | 2514.69 | 0.40 | 2129.15 | 2050.91 | 1082.63 | 2522.64 | 3000.61 |
| 960 $ms$ | 22.36 | 2999.86 | 0.35 | 2274.76 | 2207.81 | 1138.16 | 3002.95 | 3189.89 |
| 1280 $ms$ | 22.76 | 3410.10 | 0.31 | 2510.28 | 2438.99 | 1218.14 | 3411.50 | 3400.50 |
| 1600 $ms$ | 22.94 | 3577.51 | 0.29 | 2566.79 | 2433.17 | 1310.75 | 3578.60 | 3571.73 |
| 1920 $ms$ | 23.19 | 3708.04 | 0.27 | 2632.80 | 2442.63 | 1423.14 | 3709.03 | 3716.76 |
| 2240 $ms$ | 23.26 | 3870.11 | 0.25 | 2785.82 | 2564.04 | 1516.54 | 3870.45 | 3846.65 |
| 2560 $ms$ | 23.46 | 4050.40 | 0.23 | 2992.91 | 2766.25 | 1616.86 | 4050.55 | 3982.72 |
| 2880 $ms$ | 23.51 | 4236.50 | 0.22 | 3232.40 | 3019.76 | 1694.92 | 4236.58 | 4108.61 |
| 3200 $ms$ | 23.58 | 4408.96 | 0.21 | 3476.29 | 3289.87 | 1766.69 | 4409.03 | 4227.52 |
| 4800 $ms$ | 23.97 | 5161.83 | 0.18 | 4677.54 | 4654.39 | 2131.72 | 5161.83 | 4909.06 |
| 10000 $ms$ | 24.22 | 6185.38 | 0.12 | 6185.38 | 6185.38 | 3118.92 | 6185.38 | 6171.87 |
| $C$×40$ms$ | **ASR-BLEU** | **Computation-Aware Latency** | | | | | | |
| | | AL_CA | AP_CA | DAL_CA | StartOffset_CA | EndOffset_CA | LAAL_CA | ATD_CA |
| 320 $ms$ | 20.06 | 2395.05 | 0.66 | 2474.80 | 2177.62 | 811.60 | 2471.77 | 3041.25 |
| 640 $ms$ | 21.68 | 3224.90 | 0.49 | 2751.52 | 2425.16 | 1082.63 | 3231.63 | 3428.89 |
| 960 $ms$ | 22.36 | 3462.63 | 0.40 | 2625.00 | 2414.63 | 1138.16 | 3465.45 | 3410.61 |
| 1280 $ms$ | 22.76 | 3826.98 | 0.35 | 2844.11 | 2653.91 | 1218.14 | 3828.28 | 3590.27 |
| 1600 $ms$ | 22.94 | 3974.31 | 0.32 | 2945.91 | 2637.41 | 1310.75 | 3975.25 | 3782.11 |
| 1920 $ms$ | 23.19 | 4048.74 | 0.29 | 2964.03 | 2622.29 | 1423.14 | 4049.54 | 3895.00 |
| 2240 $ms$ | 23.26 | 4211.71 | 0.27 | 3124.19 | 2746.27 | 1516.54 | 4212.04 | 4035.37 |
| 2560 $ms$ | 23.46 | 4348.39 | 0.25 | 3282.05 | 2912.61 | 1616.86 | 4348.52 | 4151.34 |
| 2880 $ms$ | 23.51 | 4556.30 | 0.24 | 3521.85 | 3176.94 | 1694.92 | 4556.37 | 4291.52 |
| 3200 $ms$ | 23.58 | 4725.99 | 0.23 | 3765.53 | 3468.13 | 1766.69 | 4726.06 | 4412.92 |
| 4800 $ms$ | 23.97 | 5487.24 | 0.19 | 4921.40 | 4888.64 | 2131.72 | 5487.24 | 5086.99 |
| 10000 $ms$ | 24.22 | 6472.57 | 0.12 | 6472.57 | 6472.57 | 3118.92 | 6472.57 | 6329.43 |
| $C$×40$ms$ | **ASR-BLEU** | **Streaming Degree** | | | | | | |
| | | Num Chunks | RTF | Discontinuity Sum | Discontinuity Ave | Discontinuity Num | | |
| 320 $ms$ | 20.06 | 8.02 | 1.15 | 2141.84 | 455.29 | 5.01 | | |
| 640 $ms$ | 21.68 | 5.74 | 1.19 | 2119.97 | 600.15 | 3.81 | | |
| 960 $ms$ | 22.36 | 4.75 | 1.20 | 2010.21 | 659.49 | 3.16 | | |
| 1280 $ms$ | 22.76 | 4.02 | 1.21 | 1857.09 | 728.14 | 2.52 | | |
| 1600 $ms$ | 22.94 | 3.63 | 1.23 | 1953.86 | 880.22 | 2.16 | | |
| 1920 $ms$ | 23.19 | 3.33 | 1.25 | 2050.74 | 1068.46 | 1.88 | | |
| 2240 $ms$ | 23.26 | 3.05 | 1.27 | 2024.67 | 1226.73 | 1.60 | | |
| 2560 $ms$ | 23.46 | 2.79 | 1.28 | 1921.86 | 1326.79 | 1.36 | | |
| 2880 $ms$ | 23.51 | 2.56 | 1.29 | 1751.37 | 1357.80 | 1.15 | | |
| 3200 $ms$ | 23.58 | 2.38 | 1.30 | 1546.32 | 1298.67 | 0.99 | | |
| 4800 $ms$ | 23.97 | 1.81 | 1.37 | 539.01 | 534.01 | 0.37 | | |
| 10000 $ms$ | 24.22 | 1.00 | 1.52 | 0.00 | 0.00 | 0.00 | | |
| $C$×40$ms$ | **ASR-BLEU** | **Quality with other Metrics** | | | | | | |
| | | ASR-BLEU (with silence) | | BLASER 2.0 (Unsupervised) | BLASER 2.0 (QE) | BLASER 2.0 (Ref) | | |
| 320 $ms$ | 20.06 | 14.76 | | 0.5011 | 3.2251 | 3.0946 | | |
| 640 $ms$ | 21.68 | 16.57 | | 0.5053 | 3.2593 | 3.1567 | | |
| 960 $ms$ | 22.36 | 17.49 | | 0.5072 | 3.2722 | 3.1783 | | |
| 1280 $ms$ | 22.76 | 18.29 | | 0.5074 | 3.2767 | 3.1856 | | |
| 1600 $ms$ | 22.94 | 18.64 | | 0.5083 | 3.2812 | 3.1921 | | |
| 1920 $ms$ | 23.19 | 18.93 | | 0.5085 | 3.2842 | 3.1994 | | |
| 2240 $ms$ | 23.26 | 19.29 | | 0.5089 | 3.2866 | 3.2052 | | |
| 2560 $ms$ | 23.46 | 20.12 | | 0.5093 | 3.2908 | 3.2099 | | |
| 2880 $ms$ | 23.51 | 20.80 | | 0.5099 | 3.2943 | 3.2157 | | |
| 3200 $ms$ | 23.58 | 21.22 | | 0.5102 | 3.2957 | 3.2176 | | |
| 4800 $ms$ | 23.97 | 23.29 | | 0.5108 | 3.3017 | 3.2295 | | |
| 10000 $ms$ | 24.22 | 24.22 | | 0.5114 | 3.3075 | 3.2397 | | |

Table 10: Numerical results of StreamSpeech on CVSS-C Es→En.

| CVSS-C De→En | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $C$×40$ms$ | **ASR-BLEU** | **Latency** | | | | | | |
| | | AL | AP | DAL | StartOffset | EndOffset | LAAL | ATD |
| 320 $ms$ | 14.56 | 1687.62 | 0.46 | 1815.81 | 1758.31 | 1186.14 | 1741.47 | 2736.31 |
| 640 $ms$ | 15.83 | 2561.63 | 0.36 | 2078.04 | 2004.57 | 1327.14 | 2566.84 | 3042.11 |
| 960 $ms$ | 16.34 | 2978.14 | 0.32 | 2256.35 | 2189.65 | 1361.72 | 2980.68 | 3202.18 |
| 1280 $ms$ | 16.57 | 3276.92 | 0.29 | 2424.27 | 2341.71 | 1426.29 | 3279.19 | 3379.61 |
| 1600 $ms$ | 16.75 | 3418.49 | 0.28 | 2477.75 | 2341.77 | 1506.44 | 3420.34 | 3516.78 |
| 1920 $ms$ | 16.85 | 3568.48 | 0.26 | 2597.41 | 2426.33 | 1587.89 | 3569.64 | 3640.85 |
| 2240 $ms$ | 17.02 | 3736.77 | 0.24 | 2777.56 | 2591.56 | 1668.12 | 3737.84 | 3758.86 |
| 2560 $ms$ | 17.17 | 3904.82 | 0.23 | 2982.85 | 2800.69 | 1733.56 | 3905.40 | 3868.65 |
| 2880 $ms$ | 17.23 | 4060.73 | 0.22 | 3193.25 | 3027.93 | 1802.33 | 4061.25 | 3984.06 |
| 3200 $ms$ | 17.16 | 4219.67 | 0.21 | 3425.01 | 3282.92 | 1862.79 | 4220.07 | 4102.51 |
| 4800 $ms$ | 17.52 | 4916.03 | 0.18 | 4519.45 | 4500.36 | 2205.11 | 4916.18 | 4736.56 |
| 10000 $ms$ | 18.05 | 5741.25 | 0.12 | 5741.25 | 5741.25 | 2968.60 | 5741.25 | 5730.22 |
| $C$×40$ms$ | **ASR-BLEU** | **Computation-Aware Latency** | | | | | | |
| | | AL_CA | AP_CA | DAL_CA | StartOffset_CA | EndOffset_CA | LAAL_CA | ATD_CA |
| 320 $ms$ | 14.56 | 2558.81 | 0.59 | 2446.13 | 2135.38 | 1186.14 | 2604.99 | 3196.74 |
| 640 $ms$ | 15.83 | 3161.91 | 0.43 | 2524.26 | 2284.92 | 1327.14 | 3166.60 | 3338.35 |
| 960 $ms$ | 16.34 | 3448.73 | 0.36 | 2590.87 | 2418.56 | 1361.72 | 3450.96 | 3407.84 |
| 1280 $ms$ | 16.57 | 3677.11 | 0.33 | 2735.80 | 2552.65 | 1426.29 | 3679.01 | 3561.59 |
| 1600 $ms$ | 16.75 | 3752.73 | 0.30 | 2764.69 | 2510.66 | 1506.44 | 3754.44 | 3681.01 |
| 1920 $ms$ | 16.85 | 3883.72 | 0.28 | 2901.05 | 2597.63 | 1587.89 | 3884.82 | 3805.99 |
| 2240 $ms$ | 17.02 | 4009.48 | 0.26 | 3024.46 | 2736.61 | 1668.12 | 4010.50 | 3905.44 |
| 2560 $ms$ | 17.17 | 4174.97 | 0.25 | 3221.65 | 2952.58 | 1733.56 | 4175.52 | 4014.60 |
| 2880 $ms$ | 17.23 | 4339.98 | 0.24 | 3444.19 | 3203.36 | 1802.33 | 4340.44 | 4147.01 |
| 3200 $ms$ | 17.16 | 4484.03 | 0.23 | 3654.72 | 3448.12 | 1862.79 | 4484.39 | 4253.07 |
| 4800 $ms$ | 17.52 | 5234.16 | 0.19 | 4766.98 | 4741.92 | 2205.11 | 5234.31 | 4912.98 |
| 10000 $ms$ | 18.05 | 5977.26 | 0.13 | 5977.26 | 5977.26 | 2968.60 | 5977.26 | 5860.21 |
| $C$×40$ms$ | **ASR-BLEU** | **Streaming Degree** | | | | | | |
| | | Num Chunks | RTF | Discontinuity Sum | Discontinuity Ave | Discontinuity Num | | |
| 320 $ms$ | 14.56 | 6.85 | 1.24 | 2246.02 | 565.17 | 4.39 | | |
| 640 $ms$ | 15.83 | 4.93 | 1.26 | 2092.95 | 703.44 | 3.24 | | |
| 960 $ms$ | 16.34 | 4.15 | 1.27 | 1942.50 | 734.28 | 2.71 | | |
| 1280 $ms$ | 16.57 | 3.64 | 1.28 | 1851.17 | 800.32 | 2.25 | | |
| 1600 $ms$ | 16.75 | 3.34 | 1.30 | 1926.99 | 948.35 | 1.96 | | |
| 1920 $ms$ | 16.85 | 3.06 | 1.31 | 1922.77 | 1074.24 | 1.69 | | |
| 2240 $ms$ | 17.02 | 2.81 | 1.33 | 1832.01 | 1165.29 | 1.43 | | |
| 2560 $ms$ | 17.17 | 2.58 | 1.34 | 1695.21 | 1200.79 | 1.22 | | |
| 2880 $ms$ | 17.23 | 2.39 | 1.35 | 1531.03 | 1182.17 | 1.04 | | |
| 3200 $ms$ | 17.16 | 2.23 | 1.36 | 1339.35 | 1112.27 | 0.88 | | |
| 4800 $ms$ | 17.52 | 1.68 | 1.43 | 482.26 | 478.56 | 0.32 | | |
| 10000 $ms$ | 18.05 | 1.00 | 1.54 | 0.00 | 0.00 | 0.00 | | |
| $C$×40$ms$ | **ASR-BLEU** | **Quality with other Metrics** | | | | | | |
| | | ASR-BLEU (with silence) | | BLASER 2.0 (Unsupervised) | BLASER 2.0 (QE) | BLASER 2.0 (Ref) | | |
| 320 $ms$ | 14.56 | 10.81 | | 0.4393 | 3.0864 | 2.8424 | | |
| 640 $ms$ | 15.83 | 12.14 | | 0.4419 | 3.1041 | 2.8808 | | |
| 960 $ms$ | 16.34 | 12.79 | | 0.4435 | 3.1141 | 2.8993 | | |
| 1280 $ms$ | 16.57 | 13.35 | | 0.4436 | 3.1166 | 2.9045 | | |
| 1600 $ms$ | 16.75 | 13.49 | | 0.4441 | 3.1203 | 2.9109 | | |
| 1920 $ms$ | 16.85 | 13.87 | | 0.4445 | 3.1232 | 2.9148 | | |
| 2240 $ms$ | 17.02 | 14.34 | | 0.4448 | 3.1234 | 2.9200 | | |
| 2560 $ms$ | 17.17 | 14.84 | | 0.4447 | 3.1234 | 2.9217 | | |
| 2880 $ms$ | 17.23 | 15.31 | | 0.4450 | 3.1267 | 2.9254 | | |
| 3200 $ms$ | 17.16 | 15.62 | | 0.4450 | 3.1253 | 2.9248 | | |
| 4800 $ms$ | 17.52 | 17.09 | | 0.4459 | 3.1311 | 2.9361 | | |
| 10000 $ms$ | 18.05 | 18.05 | | 0.4469 | 3.1394 | 2.9507 | | |

Table 11: Numerical results of StreamSpeech on CVSS-C De→En.