End-to-End Simultaneous Speech Translation with Differentiable Segmentation

Shaolei Zhang 1,2, Yang Feng 1,2

1 Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
2 University of Chinese Academy of Sciences, Beijing, China
{zhangshaolei20z, fengyang}@ict.ac.cn

Abstract

End-to-end simultaneous speech translation (SimulST) outputs translation while receiving the streaming speech inputs (a.k.a. streaming speech translation), and hence needs to segment the speech inputs and then translate based on the current received speech. However, segmenting the speech inputs at unfavorable moments can disrupt the acoustic integrity and adversely affect the performance of the translation model. Therefore, learning to segment the speech inputs at those moments that are beneficial for the translation model to produce high-quality translation is the key to SimulST. Existing SimulST methods, either using the fixed-length segmentation or external segmentation model, always separate segmentation from the underlying translation model, where the gap results in segmentation outcomes that are not necessarily beneficial for the translation process. In this paper, we propose Differentiable Segmentation (DiSeg) for SimulST to directly learn segmentation from the underlying translation model. DiSeg turns hard segmentation into differentiable through the proposed expectation training, enabling it to be jointly trained with the translation model and thereby learn translation-beneficial segmentation. Experimental results demonstrate that DiSeg achieves state-of-the-art performance and exhibits superior segmentation capability.

1 Introduction

End-to-end simultaneous speech translation (SimulST) (Fügen et al., 2007; Oda et al., 2014; Ren et al., 2020; Zeng et al., 2021; Zhang et al., 2022a) outputs translation when receiving the streaming speech inputs, and is widely used in real-time scenarios such as international conferences, live broadcasts and real-time subtitles. Compared with the offline speech translation waiting for the complete speech inputs (Weiss et al., 2017; Wang et al., 2020), SimulST needs to segment the streaming speech inputs and synchronously translate based on the current received speech, aiming to achieve high translation quality under low latency (Hamon et al., 2009; Cho and Esipova, 2016; Ma et al., 2020b; Zhang and Feng, 2022c).

However, it is non-trivial to segment the streaming speech inputs as the speech always lacks explicit boundary (Zeng et al., 2021), and segmenting at unfavorable moments will break the acoustic integrity and thereby drop the translation performance (Dong et al., 2022). Therefore, the precise segmentation of streaming speech is the core challenge of SimulST task (Zhang et al., 2022a). To ensure that the speech representations derived from the segmentation results can produce high-quality translation, SimulST model should learn a translation-beneficial segmentation from the underlying translation model.

Existing SimulST methods, involving fixed and
adaptive, always fail to learn the segmentation directly from the underlying translation model. The fixed method divides the streaming inputs based on the equal length, e.g., 280ms per segment (Ma et al., 2020b, 2021; Nguyen et al., 2021), as shown in Figure 1(a). Such methods completely ignore the translation model and always break the acoustic integrity (Dong et al., 2022). The adaptive method dynamically decides the segmentation and thereby achieves better SimulST performance, as shown in Figure 1(b). However, previous adaptive methods often use the external segmentation model (Ma et al., 2020b; Zhang et al., 2022a) or heuristic detector (Zeng et al., 2021; Chen et al., 2021; Dong et al., 2022) for segmentation, which leave a gap between the segmentation and translation model. This gap hinders learning segmentation directly from the translation model (Arivazhagan et al., 2019), hence making them difficult to get segmentation results that are most beneficial to translation quality.

Under these grounds, we aim to integrate the segmentation into translation model, and thereby directly learn segmentation from the underlying translation model, as shown in Figure 1(c). To this end, we propose Differentiable Segmentation (DiSeg) for SimulST, which can be jointly trained with the underlying translation model. DiSeg employs a Bernoulli variable to indicate whether the streaming speech inputs should be segmented or not. Then, to address the issue that hard segmentation precludes back-propagation (i.e., learning) from the underlying translation model, we propose an expectation training to turn the segmentation into differentiable. Owing to powerful segmentation, DiSeg can handle simultaneous and offline speech translation through a unified model. Experiments show that DiSeg achieves state-of-the-art performance on SimulST, while also delivering high-quality offline speech translation.

2 Background

Offline Speech Translation The corpus of speech translation task is always denoted as the triplet \( \mathcal{D} = \{ (s, x, y) \} \), where \( s = (s_1, \cdots, s_{|s|}) \) is source speech, \( x = (x_1, \cdots, x_{|x|}) \) is source transcription and \( y = (y_1, \cdots, y_{|y|}) \) is target translation. The mainstream speech translation architecture often consists of an acoustic feature extractor and a translation model following (Nguyen et al., 2020). Acoustic feature extractor extracts speech features \( a = (a_1, \cdots, a_{|a|}) \) from source speech \( s \), which is often realized by a pre-trained acoustic model (Baevski et al., 2020). Then, the translation model, realized by a Transformer model (Vaswani et al., 2017), generates \( y \) based on all speech features \( a \). During training, existing methods always improve speech translation performance through multi-task learning (Anastasopoulos and Chiang, 2018; Tang et al., 2021a,b), including speech translation, automatic speech recognition (ASR) and machine translation (MT) (add a word embedding layer for MT task), where the learning objective \( L_{mtl} \) is:

\[
L_{mtl} = L_{st} + L_{asr} + L_{mt},
\]

where \( L_{st}, L_{asr} \) and \( L_{mt} \) are the cross-entropy loss of pairs \( s \rightarrow y, s \rightarrow x \) and \( x \rightarrow y \), respectively.

Simultaneous Translation (SimulST) Unlike offline speech translation, SimulST needs to decide when to segment the inputs and then translate based on the received speech features (Ren et al., 2020; Ma et al., 2020b). Since the decisions are often made based on the speech features after downsampling, we use \( g(t) \) to denote the number of speech features when the SimulST model translates \( y_t \), where speech features \( a_{\leq g(t)} \) are extracted from the current received speech \( \hat{s} \). Then, the probability of generating \( y_t \) is \( p(y_t \mid \hat{s}, y_{\leq t}) \). How to decide \( g(t) \) is the key of SimulST, which should be beneficial for the translation model to produce the high-quality translation.

3 Method

We propose differentiable segmentation (DiSeg) to learn segmentation directly from the translation model, aiming to achieve translation-beneficial segmentation. As shown in Figure 2, DiSeg predicts a Bernoulli variable to indicate whether to segment, and then turns the hard segmentation into differentiable through the proposed expectation training, thereby jointly training segmentation with the translation model. We will introduce the segmentation, training and inference of DiSeg following.

3.1 Segmentation

To segment the streaming speech inputs, DiSeg predicts a Bernoulli variable 0/1 for each speech feature, corresponding to waiting or segmenting. Specifically, for the speech feature \( a_i \), DiSeg predicts a Bernoulli segmentation probability \( p_i \), corresponding to the probability of segmenting the speech at \( a_i \). Segmentation probability \( p_i \) is calculated through a feed-forward network (FFN) fol-
lowed by a sigmoid activation, and then $p_i$ is used to parameterize the Bernoulli variable $b_i$:

$$p_i = \text{Sigmoid}(\text{FFN}(a_i)),$$  \hspace{1cm} (2)

$$b_i \sim \text{Bernoulli}(p_i).$$ \hspace{1cm} (3)

If $b_i = 1$, DiSeg segments the streaming speech at $a_i$; If $b_i = 0$, DiSeg waits for more inputs. In inference, DiSeg sets $b_i = 1$ if $p_i \geq 0.5$, and sets $b_i = 0$ if $p_i < 0.5$ (Raffel et al., 2017).

**Segmented Attention** After segmenting the speech features, we propose segmented attention for the encoder of the translation model, which is an attention mechanism between uni-directional and bi-directional attention. In segmented attention, each speech feature can focus on the features in the same segment and the previous segments (i.e., bi-directional attention within a segment, uni-directional attention between segments), as shown in Figure 3(a). In this way, segmented attention can not only satisfy the requirement of encoding streaming inputs in SimulST task (i.e., the characteristic of uni-directional attention) (Elbayad et al., 2020; Zeng et al., 2021), but also capture more comprehensive context representations of segments (i.e., the advantage of bi-directional attention).

### 3.2 Expectation Training

Hard segmentation based on Bernoulli variable $b_i$ precludes back-propagation (i.e., learning) from the translation model to the segmentation probability $p_i$ during training. To address this, we propose expectation training to turn segmentation into differentiable. In expectation training, we first constrain the number of segments, and then learn segmentation from the translation model at both acoustic and semantic levels, which are all trained in expectation via the segmentation probability $p_i$.

#### Learning Segment Number

To avoid too many segments breaking the acoustic integrity or too few segments degenerating the model into offline translation, we need to constrain the total number of segments. Intuitively, the source speech should be divided into $K$ segments, where $K$ is the number of words in the source transcription, so that each speech segment can correspond to a complete word. To this end, we apply $\mathcal{L}_{num}$ to constrain the expected segment number to be $K$. In particular, in order to prevent excessive segmentation on consecutive silent frames, we also encourage only one segmentation in several consecutive speech frames. Therefore, $\mathcal{L}_{num}$ is calculated as:

$$\mathcal{L}_{num} = \left\| \sum_{i=1}^{\lfloor \frac{|a|}{K} \rfloor} p_i - K \right\|_2 + \left\| \sum_{i=1}^{\lfloor \frac{|a|}{K} \rfloor} \text{MaxPool} \left( p_i, \left\lfloor \frac{|a|}{K} \right\rfloor \right) - K \right\|_2,$$  \hspace{1cm} (4)

where $\sum_{i=1}^{\lfloor \frac{|a|}{K} \rfloor} p_i$ is the expected segment number and $\text{MaxPool} (\cdot)$ is the max pooling operation with kernel size of $\lfloor \frac{|a|}{K} \rfloor$.

To make the effect of $p_i$ in expectation training match $b_i$ in the inference, we hope that $p_i \approx 0$ or $p_i \approx 1$ and thereby make $p_i \approx b_i$. To achieve this, we aim to encourage the discreteness of segmentation probability $p_i$ during training. Following Salakhutdinov and Hinton (2009); Foerster et al. (2016); Raffel et al. (2017), a straightforward and efficient method is adding a Gaussian noise before the sigmoid activation in Eq.(2). Formally, in expectation training, Eq.(2) is rewritten as:

$$p_i = \text{Sigmoid}(\text{FFN}(a_i) + \mathcal{N}(0, n)),$$  \hspace{1cm} (6)

where $\mathcal{N}(0, n)$ is a Gaussian noise with 0 as mean and $n$ as variance. Noise is only applied in training.
Learning Segmentation at Acoustic Level

A good segmentation should avoid breaking acoustic integrity and benefit the underlying translation model. As mentioned in Sec.3.1, the encoder of the translation model applies the segmented attention to model the correlation between speech features and get the source representations. Correspondingly, we propose expected segmented attention to turn the hard segmentation into differentiable during training, thereby directly learning translation-beneficial segmentation from the translation model.

In segmented attention during inference, speech feature $a_i$ can only pay attention to feature $a_j$ that locates in the same segment or previous segments, and mask out the rest features. In expected segmented attention, to enable back-propagation, we introduce the probability that $a_i$ can pay attention to $a_j$ instead of the hard segmentation, denoted as $\beta_{i,j}$. As shown in Figure 3(b), $\beta_{i,j}$ measures the probability that $a_j$ locates in the same segment as $a_i$ or in the segment before $a_i$, calculated as:

$$
\beta_{i,j} = \begin{cases} 
\prod_{l=i}^{j-1} (1 - p_l), & \text{if } i < j \\
1, & \text{if } i \geq j
\end{cases},
$$

(7)

If $a_j$ lags behind $a_i$, the premise that $a_i$ and $a_j$ are in the same segment is that no segmentation is between $a_i$ and $a_{j-1}, i.e., \prod_{l=i}^{j-1} (1 - p_l)$. If $a_j$ is before $a_i$, $a_i$ can necessarily focus on $a_j$. Then, $\beta_{i,j}$ is multiplied with the original soft attention $\alpha_{i,j}$, and normalized to get the final attention $\gamma_{i,j}$:

$$
\tilde{\gamma}_{i,j} = \alpha_{i,j} \times \beta_{i,j},
$$

(8)

$$
\gamma_{i,j} = \frac{\tilde{\gamma}_{i,j}}{\sum_{l=1}^{|a|} \tilde{\gamma}_{i,l}}.
$$

(9)

Finally, $\gamma_{i,j}$ is used to calculate the context vector. Owing to the expected segmented attention, $p_i$ can be jointly trained with the translation model via cross-entropy loss $L_{mtl}$. Specifically, if the underlying translation model prefers to let $a_i$ pay attention to the subsequent $a_j$ for better source representation (i.e., a large $\gamma_{i,j}$), the probability $\beta_{i,j}$ that they are in the same segment will be increased, which teaches DiSeg to prevent segmenting between $a_i$ and $a_j$. In this way, DiSeg can avoid breaking acoustic integrity and learn the segmentation that is beneficial to the translation model.

Learning Segmentation at Semantic Level

Besides encouraging the related speech features to locate in the same segment via expected segmented attention, we aim to further learn segmentation at the semantic level. In the multi-task learning framework, the transcription $x$ is monotonically corresponding to the source speech, so transcription is a good choice to provide semantic supervision for segmentation. However, there is a significant gap between transcription representations and speech features in sequence length (Liu et al., 2020; Zeng et al., 2021), so how to align them is key challenge for semantic supervision. Fortunately, the proposed differentiable segmentation divides the speech into $K$ segments, where $K$ is also the word number in transcription. Therefore, both speech and transcription sequences can be mapped to the sequence of length $K$ and accordingly reach an agreement on the corresponding representation to achieve semantic supervision, as shown in Figure 4.

For transcription $x$, since it is a sequence of subwords after tokenization (Kudo and Richardson, 2018), we introduce a subword-to-word map to get the representation of the whole word. Given the text embedding $e = \text{Emb}(x)$ of subwords, the whole word representation is the average pooling result on the embedding of all subwords it contains. Formally, the representation $f^k_i$ of the $k$th word that consists of subwords $x[l_k : r_k]$ is calculated as:

$$
f^k_i = \frac{1}{r_k - l_k + 1} \sum_{l=l_k}^{r_k} e_i.
$$

(10)

For speech, the segment representations also need to be differentiable to segmentation (i.e., $p_i$), thereby enabling DiSeg to learn segmentation at semantic level. To this end, we propose an expected
Algorithm 1: Wait-seg Policy for DiSeg

Input: streaming speech inputs s, lagging segments k
Output: target outputs ˆ y

Initialization: ˆ y0 = (BOS), target index t = 1,
current received speech ˆ s = [ ]

1 while ˆ yt−1 ≠ (EOS) do
2 Extract speech features {ai}i=1 from ˆ s;
3 Predict segmentation {bi}i=1;
4 if i=1 bi ≥ t+k−1 or ˆ s = s then // Translate
5 ˆ yt ← ˆ yt−1; t ← t + 1;
6 else // Wait
7 ˆ s ← ˆ s + s.read();
8 return ˆ y;

The specific inference is shown in Algorithm 1.

To keep the training and inference matching, we also apply the wait-seg policy during training via the proposed wait-seg decoder. When translating ˆ yt, wait-seg decoder will mask out the speech features ˆ ai that i > g(t; k) (Ma et al., 2019). Accordingly, we introduce multi-task training and multi-latency training to enhance DiSeg performance.

Multi-task Training Since we adopt the multi-task learning framework (refer to Sec.2), ASR and MT tasks should also adapt to DiSeg. Specifically, ASR task applies the same segmentation and policy (i.e., decoder) as the SimulST task, as both their inputs are speech. For the MT task, since the segment in the speech corresponds to the word in the transcription, MT task applies a uni-directional encoder and wait-k policy (i.e., decoder). Note that parameters of encoder and decoder are shared among various tasks.

Multi-latency Training To enhance the DiSeg performance under multiple latency, we randomly sample k from [1, K] between batches during training (Elbayad et al., 2020). In inference, DiSeg only needs one model to complete SimulST under multiple arbitrary latency (Zhang and Feng, 2021c), including offline speech translation (the latency is the complete speech duration). In this way, DiSeg develops a unified model that can handle both offline and simultaneous speech translation.

4 Experiments

4.1 Datasets

We conduct experiments on two end-to-end simultaneous translation benchmarks, MuST-C2 English → German (En→De, 234K pairs) and English → Spanish (En→Es, 270K pairs) (Di Gangi et al., 2019). We use dev as the validation set (1423 pages of source data).
pairs for En→De, 1316 pairs for En→Es) and use t-st-COMMON as the test set (2641 pairs for En→De, 2502 pairs for En→Es), respectively. For speech, we use the raw 16-bit 16kHz mono-channel audio wave. For text, we use SentencePiece (Kudo and Richardson, 2018) to generate a unigram vocabulary of size 10000, sharing between languages.

4.2 System Settings

We conduct experiments on the following systems.

**Offline** Offline speech translation, which waits for the complete speech inputs and then translates (bi-directional attention and greedy search).

**Wait-k** Wait-k policy (Ma et al., 2019) with fixed segmentation of speech (Ma et al., 2020b), which translates a word every 280 ms.

**Wait-k-Stride-n** A variation of wait-k policy (Zeng et al., 2021), which translates every \( n \times 280 \) ms. We set \( n = 2 \) following their best result.

**MMA** Monotonic multihead attention (Ma et al., 2020c), which is adapted to SimulST by dividing speech into segments of equal length, i.e., 120 ms, 200 ms and 280 ms · · · (Ma et al., 2020b).

**MMA-CMDR** MMA with cross-modal decision regularization (Zaidi et al., 2022), which leverages the transcription to improve the decision of MMA.

**SimulSpeech** Segmentation based on word detector (Ren et al., 2020), which also uses two knowledge distillations to improve the performance.

**SH** Synchronized ASR-assisted SimulST (Chen et al., 2021), which uses the shortest hypothesis in ASR results to indicate the word number in speech.

**RealTrans** A convolutional weighted-shrinking Transformer (Zeng et al., 2021), which detects the word number in the streaming speech and then decodes via the wait-k-stride-n policy.

**MoSST** Monotonic-segmented streaming speech translation (Dong et al., 2022), which uses the integrate-and-firing method (Dong and Xu, 2020) to segment the speech based on the cumulative acoustic information.

**ITST** Information-transport-based policy for SimulST (Zhang and Feng, 2022b), which quantifies the transported information from source to target, and then decides whether to translate according to the accumulated received information.

**MU-ST** Segmentation based on the meaning unit (Zhang et al., 2022a), which trains an external segmentation model based on the constructed data, and uses it to decide when to translate.

**DiSeg** The proposed method in Sec.3.

All implementations are adapted from Fairseq Library (Ott et al., 2019). We use a pre-trained Wav2Vec2.0 (Baevski et al., 2020) as the acoustic feature extractor, and use a standard Transformer-Base (Vaswani et al., 2017) as the translation model. For evaluation, we apply SimulEval (Ma et al., 2020a) to report SacreBLEU (Post, 2018) for translation quality and Average Lagging (AL) (Ma et al., 2019) for latency. AL measures the average dura-
tion (ms) that target outputs lag behind the speech inputs. The calculation refer to Appendix D.

4.3 Main Results

We compare DiSeg and previous SimulST methods in Figure 5, where we only train a single DiSeg model and adjusted the lagging number $k$ during the inference process to show the translation quality under different latency. Remarkably, DiSeg outperforms strong baselines under all latency and achieves state-of-the-art performance. Compared with fixed methods, such as Wait-$k$ and MMA (Ma et al., 2020b), DiSeg can dynamically decide segmentation according to the inputs instead of equal-length segmentation, which avoids breaking the acoustic integrity and thus achieves notable improvements. Compared with adaptive methods, including the state-of-the-art MU-ST, DiSeg also performs better. In previous adaptive methods, regardless of RealTrans, SH and SimulSpeech detecting the word number (Zeng et al., 2021; Chen et al., 2021; Ren et al., 2020), MoSST and ITST comparing the acoustic information with a threshold (Dong et al., 2022; Zhang and Feng, 2022b), or MU-ST training an external segmentation model (Zhang et al., 2022a), the final translation results is always non-differentiable to the segmentation, which hinders learning segmentation directly from the translation model. The proposed DiSeg turns the segmentation into differentiable, hence can learn translation-beneficial segmentation directly from the translation model, thereby achieving better performance. Furthermore, unlike the previous methods using uni-directional (e.g., RealTrans and ITST) or bi-directional attention (e.g., MU-ST and MoSST), the proposed segmented attention can not only encode streaming inputs but also get comprehensive segment representations (Zhang et al., 2021). In particular, DiSeg achieves comparable performance with the offline model when lagging 2300ms on En→De and 3000ms on En→Es, which is attributed to the improvements on translation quality brought by the segmented attention.

5 Analysis

We conduct extensive analyses to study the effectiveness and specific improvements of DiSeg. Unless otherwise specified, all the results are reported on MuST-C En→De test set.

5.1 Ablation Study

Discreteness of Segmentation Probability To make expectation training more suitable for inference, we encourage the discreteness of segment probability via introducing Gaussian noise $N(0, n)$ in Eq.(6). We compare the effect of discreteness in Figure 6(a), where appropriately encouraging discreteness effectively enhances expectation training, thereby improving DiSeg performance under low latency. However, too much noise will affect translation quality, especially under high latency, which is consistent with Arivazhagan et al. (2019).

Number of Segments In DiSeg, we constrain the number of segments to be the word number $K$ in the transcription rather than subword. To verify the effectiveness of segmentation granularity, we compare different segment numbers in Figure 6(b), noting that $L_{ctr}$ is also changed to be computed by subword embedding accordingly. Segmentation on word granularity is significantly better than subword granularity, mainly because many subwords are actually continuous and related in the speech, and segmentation at the word granularity can better preserve the acoustic integrity (Dong et al., 2022).

Learning at Acoustic and Semantic Levels DiSeg learns segmentation at the acoustic and semantic levels, so we show the effectiveness of acoustic and semantic learning in Figure 6(c). The results demonstrate that both acoustic and semantic learning play the important role in SimulST per-
performance. Specifically, acoustic learning encourages related speech features to be in the same segment through expected segment attention, where ensuring acoustic integrity is more important for SimulST under low latency, thereby achieving an improvement of 2 BLEU (AL = 1500 ms). Semantic learning supervises the segment representations through the word representations in the transcription, which helps the segment representations to be more conducive to the translation model (Ye et al., 2022), thereby improving the translation quality.

**Effect of Wait-seg Decoder** DiSeg introduces wait-seg decoder to learn wait-seg policy. As shown in Figure 6(d), compared with full decoder that can focus on all speech features, wait-seg decoder enhances DiSeg’s ability to translate based on partial speech (Ma et al., 2019) and thus achieves significant improvements during inference.

### 5.2 Segmented Attention on Offline ST

How to encode streaming inputs is an important concern for SimulST (Zhang et al., 2021), where offline translation uses bi-directional attention to encode the complete source input and existing SimulST methods always apply uni-directional attention (Zeng et al., 2021; Zhang and Feng, 2022b). DiSeg applies segmented attention, which consists of bi-directional attention within a segment and uni-directional attention between segments. To study the modeling capacity of segmented attention, we compare the performance of uni-directional attention, bi-directional attention and DiSeg (segmented attention) on offline speech translation in Table 1. DiSeg is a unified model that can handle both simultaneous speech translation and offline speech translation together. Therefore, we employ the same model as SimulST to complete offline speech translation, while only setting \( k = \infty \) and applying beam search during inference.

Uni- and bi-directional attention achieve similar performance in greedy search, which is consistent with Wu et al. (2021), while bi-directional attention performs better in beam search due to more comprehensive encoding. Owing to learning the translation-beneficial segmentation, DiSeg can outperform uni-/bi-directional attention on both greedy and beam search when only applying bi-directional attention within segments. Furthermore, when removing \( L_{ctr} \), segmented attention also achieves comparable performance to bi-directional attention, providing a new attention mechanism for future streaming models. Appendix C.2 gives visualization and more analyses of segmented attention.

### 5.3 Segmentation Quality

To explore the segmentation quality of DiSeg, we conduct experiments on speech segmentation task (Demuynck and Laureys, 2002) with the annotated Buckeye corpus\(^8\) (Pitt et al., 2005). Table 2 shows the segmentation performance of DiSeg and strong baselines (Kamper et al., 2017b,a; Kamper and van Nierkerk, 2020; Bhati et al., 2022; Fuchs et al., 2022), and the metrics include precision (P), recall (R), F1 score, over-segmentation (OS) and R-value (refer to Appendix B for the calculation). The results show that DiSeg achieves better segmentation performance and \( L_{ctr} \) can improve the segmentation quality by 1% score. More importantly, DiSeg achieves an OS score close to 0, demonstrating that DiSeg can get the appropriate number of segments, thereby avoiding too many or too few segments affecting SimulST performance (Zhang et al., 2022b). We will analyze the number of segments following.

### 5.4 Segmentation Quantity

In training, we constrain the number of segments to be the word number in the transcription. To verify its effectiveness, we count the difference between the segment number and the word number during inference (i.e., \#Segments—\#Words) in Figure 7.

---

\( ^8 \)https://buckeyecorpus.osu.edu
Compared with the previous work considering that 280 ms corresponds to a word on average (Ma et al., 2020b; Zaidi et al., 2022), DiSeg can get a more accurate number of segments, where the difference between the segment number and the word number is less than 2 in 70% of cases. Besides, as reported in Table 2, the OS score on the automatic speech segmentation task also demonstrates that DiSeg can achieve an appropriate number of segments. Therefore, constraining the expected segment number in expectation training is effective to control the number of segments.

5.5 Adapting Multi-task Learning to SimulST

During training, we adjust ASR task (segmented encoder + wait-seg decoder) and MT task (uni-encoder + wait-k decoder) in multi-task learning to adapt to DiSeg, so we verify the effectiveness of the adaptation in Table 3. In the proposed adaptation #1, since a speech segment corresponds to a word, both uni-encoder and wait-k decoder in MT task are fully compatible with the segmented attention and wait-seg decoder in SimulST task, thereby performing best. Both using bi-encoder (i.e., setting #5-7) or full decoder (i.e., setting #2-4) in ASR and MT tasks will affect DiSeg performance, and the performance degradation caused by the encoder mismatch is more serious. In general, ASR and MT tasks should be consistent and compatible with the SimulST task when adapting multi-task learning.

6 Related Work

Early SimulST methods segment speech and then use the cascaded model (ASR+MT) to translate each segment (Fügen et al., 2007; Yarmohammadi et al., 2013; Rangarajan Sridhar et al., 2013; Zhang and Feng, 2023; Guo et al., 2023). Recent end-to-end SimulST methods fall into fixed and adaptive.

7 Conclusion

In this study, we propose differentiable segmentation (DiSeg) for simultaneous speech translation to directly learn segmentation from the underlying translation model. Experiments show the superiority of DiSeg in terms of SimulST performance, attention mechanism and segmentation quality.

Future researches will delve into the untapped potential of differentiable segmentation in such streaming models and long sequence modeling, thereby reducing feedback latency or computational cost without compromising performance.
Acknowledgements

We thank all the anonymous reviewers for their insightful and valuable comments. This work was supported by National Key R&D Program of China (NO. 2018AAA0102502).

Limitations

In this study, we propose differentiable segmentation to learn how to segment speech from the underlying translation model, and verify its effectiveness on simultaneous speech translation. However, since it can be jointly trained with the underlying task (sequence-to-sequence task), differentiable segmentation is not limited to the SimulST task, but can be generalized to more streaming/online tasks, such as streaming automatic speech recognition (streaming ASR), simultaneous machine translation (SiMT), real-time text-to-speech synthesis (real-time TTS), online tagging and streaming parsing. Given that there may be some task-specific differences between various tasks, this work only focuses on the differentiable segmentation in the SimulST task, and we leave the study of how to apply differentiable segmentation to other streaming tasks into our future work.

References


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


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


Shaolei Zhang and Yang Feng. 2022a. Gaussian multi-head attention for simultaneous machine translation. In Findings of the Association for Computational


A Dynamic Programming for Expected Feature-to-Segment Mapping

In Sec.3.2, we propose the expected feature-to-segment map to get the expected segment representations while keeping segment representations differentiable to the segmentation. Expected feature-to-segment map calculates the probability \( p(a_i \in \text{Seg}_k) \) that the speech feature \( a_i \) belongs to the \( k^{th} \) segment \( \text{Seg}_k \), and then gets the expected segment representation by weighting all speech features \( a_i \) with \( p(a_i \in \text{Seg}_k) \).

Given segmentation probability \( p_i \), we calculate \( p(a_i \in \text{Seg}_k) \) via dynamic programming. Whether \( a_i \) belongs to the \( k^{th} \) segment depends on which segment that speech feature \( a_{i-1} \) is located in, consisting of 3 situations:

- \( a_{i-1} \in \text{Seg}_{k-1} \): If DiSeg segments at feature \( a_{i-1} \) (with probability \( p_{i-1} \)), then \( a_i \) belongs to \( \text{Seg}_k \);
- \( a_{i-1} \notin \text{Seg}_k \): If DiSeg does not segment at feature \( a_{i-1} \) (with probability \( 1 - p_{i-1} \)), then \( a_i \) belongs to \( \text{Seg}_k \);
- Others: \( a_i \) can not belong to \( \text{Seg}_k \) anyway, because the feature-to-segment mapping must be monotonic.

By combining these situations, \( p(a_i \in \text{Seg}_k) \) is calculated as:

\[
p(a_i \in \text{Seg}_k) = p(a_{i-1} \in \text{Seg}_{k-1}) \times p_{i-1} + p(a_{i-1} \notin \text{Seg}_k) \times (1 - p_{i-1}).
\]

(16)

For the initialization, \( p(a_1 \in \text{Seg}_k) \) is calculated as:

\[
p(a_1 \in \text{Seg}_k) = \begin{cases} 1 & \text{if } k = 1 \\ 0 & \text{if } k \neq 1 \end{cases}.
\]

(17)

where the first feature inevitably belongs to the first segment. Since we constrained the number of segments to be \( K \) during training, we truncate \( p(a_i \in \text{Seg}_k) \) at \( p(a_i \in \text{Seg}_K) \).

B Metrics of Word Segmentation Task

In Sec.5.3, we evaluate the segmentation quality of DiSeg on the automatic speech segmentation task (Demuynck and Laureys, 2002; Sakran et al., 2017), and here we give the specific calculation of metrics for the speech segmentation task.

Precision (P), recall (R) and the corresponding F1 score are used to measure whether the segmentation position is correct compared with the ground-truth segmentation. Over-segmentation (OS) (Petek et al., 1996) measures whether the number of segments generated by the model is accurate, calculated as:

\[
\text{OS} = \frac{R}{P} - 1,
\]

(18)

where OS = 0 means that the number of segments is completely accurate, a larger OS means more segments, and a smaller OS means fewer segments. Since a large number of segments is easy to obtain a high recall score while a poor OS score, a more robust metric R-value (Räsänen et al., 2009) is proposed to measure recall score and OS score together. R-value is calculated as:

\[
\text{R-value} = 1 - \frac{|r_1| + |r_2|}{2},
\]

(19)

where

\[
r_1 = \sqrt{(1 - R)^2 + \text{OS}^2},
\]

(20)

\[
r_2 = \frac{-\text{OS} + R - 1}{\sqrt{2}}.
\]

(21)

A larger R-value indicates better segmentation quality, and the only way to achieve a perfect R-value is to get a perfect recall score (i.e., \( R = 1 \)) and a perfect OS score (i.e., \( OS = 0 \)).

C Extended Analyses

C.1 Effectiveness of Contrastive Learning

During expectation training, we use the word representations to supervise the segment representations via contrastive learning \( L_{ctr} \) (Sohn, 2016). To
verify its effect, we compared the performance of applying contrastive learning loss $L_{ctr}$ and some other loss functions for semantic supervision in Figure 8, including

- $L_{ctr}$: reduce the cosine similarity between the expected segment representation $f_k^s$ and the corresponding word representation $f_k^t$ and meanwhile separates $f_k^s$ with the rest of the word representations;

- $L_{cos}$: reduce the cosine similarity between the expected segment representation $f_k^s$ and the corresponding word representation $f_k^t$;

- $L_2$: reduce the $L_2$ distance between the expected segment representation $f_k^s$ and the corresponding word representation $f_k^t$.

The results show that the cosine similarity $L_{cos}$ is better than the $L_2$ distance to measure the difference between the segment representation and the word representation (Le-Khac et al., 2020; Ye et al., 2022), and the contrastive learning $L_{ctr}$ further improves DiSeg performance by introducing negative examples. In particular, since $L_{cos}$ and $L_2$ loss fails to separate the representation of the segment and non-corresponding words, it is easy to cause the segment corresponds to more words or fewer words, which can still reduce $L_{cos}$ or $L_2$ loss but is not conducive to learning the precise segmentation. By making positive pairs (i.e., segment and the corresponding word) attracted and negative pairs (i.e., segment and those non-corresponding words) separated (Sohn, 2016), contrastive learning $L_{ctr}$ can learn more precise segmentation boundaries and thereby achieve better performance.

C.2 Visualization of Segmented Attention

We visualize the proposed segmented attention, the previous uni-directional attention and bi-directional attention in Figure 9, 10 and 11, including the speech with various lengths.

Comprehensive Compared with bi-directional attention, uni-directional attention obviously loses some information from the subsequent speech features (Zhang et al., 2021; Zhang and Feng, 2022d; Iranzo Sanchez et al., 2022). Segmented attention applies bidirectional attention within a segment and thereby can get a more comprehensive representation. In Figure 9, compared with uni-directional attention, segmented attention is more similar with bi-directional attention in attention distribution.

Precise DiSeg learns the translation-beneficial segmentation through the proposed expected segmented attention (refer to Eq.(7)), which encourages the model to segment the inputs at the feature $a_i$ if $a_i$ does not need to pay attention to subsequent features. As shown in Figure 9(b), 10(b) and 11(b), DiSeg can learn precise segmentation and ensure the acoustic integrity in each segment. In particular, the segmentation shown in Figure 9(b) and 10(b) almost guarantees that each speech segment corresponds to a word in the transcription. Note that the last segment in Figure 9(b) and 10(b) often corresponds to silence at the end of speech. Since DiSeg learns segmentation without labeled segmentation/alignment data, the proposed segmented attention can be applied to more streaming tasks.

Concentrate The issue of attention dispersion caused by long speech is one of the major challenges for speech modeling (Yang et al., 2020; Liang et al., 2021; Valentini-Botinhao and King, 2021; Zheng et al., 2020). As shown in Figure 11(a) and 11(c), both uni-directional and bi-directional attention tends to become scattered when dealing with long speech, and each feature can only get a very small amount of attention weight (e.g., the maximum attention weight in Figure 11(c) is 0.01.), which affects the modeling capability of the attention mechanism (Vig and Belinkov, 2019; Ding et al., 2019; Valentini-Botinhao and King, 2021). Segmented attention applies bi-directional attention within a segment and uni-directional attention between segments, which naturally introduces locality to attention, thereby effectively mitigating the issue of attention dispersion (Luong et al., 2015; Yang et al., 2018; Liang et al., 2021; Zhang and Feng, 2021b; Zheng et al., 2020). Specifically, as shown in Figure 10(b), segmented attention can be concentrated in the segment and pay more attention to the surrounding features. As shown in Figure 11(b), although the sequence of speech features is extremely long, segmented attention also can focus on the features in each segment (e.g., a clear attention distribution can be found inside each segment in Figure 11(b), and the maximum attention weight is 0.47.). Therefore, segmented attention provides a solution to enhance locality in long speech modeling.

C.3 Case Study

We visualize the simultaneous translation process of DiSeg on simple and hard cases in Figure 12.
Figure 9: Visualization of segmented attention and uni-/bi-directional attention on short speech Ted_1160_70. Duration: 0.31 s; Transcription: ‘Thank you.’ Three speech segments in the segmented attention correspond to ‘Thank’, ‘you’ and silence in the transcription respectively. The color shade indicates the attention weight, and the blank area indicates that the attention is masked out.

Figure 10: Visualization of segmented attention and uni-/bi-directional attention on medium length speech Ted_1171_11. Duration: 1.69 s; Transcription: ‘That’s about a 15-foot boat.’ Seven speech segments in the segmented attention correspond to ‘That’s’, ‘about’, ‘a’, ‘15’, ‘foot’, ‘boat’, and silence in the transcription respectively. The color shade indicates the attention weight, and the blank area indicates that the attention is masked out.

Figure 11: Visualization of segmented attention and uni-/bi-directional attention on extremely long speech Ted_1104_16. Duration: 17.09 s; Transcription: ‘Let me now introduce you to eLEGS that is worn by Amanda Boxtel that 19 years ago was spinal cord injured, and as a result of that she has not been able to walk for 19 years until now.’. The color shade indicates the attention weight, and blank area indicates that the attention is masked out.
Figure 12: Case study of DiSeg. The horizontal direction indicates when the model outputs the target word with the streaming speech inputs. Red lines indicate where DiSeg decides to segment the speech inputs, and gray lines indicate the equal-length segmentation of 280 ms. The green areas indicate the alignments between the transcription and the speech generated by an offline alignment tool.

Horizontally, the position of the word in outputs is the moment when it is translated, corresponding to the speech inputs. Red lines indicate where DiSeg decides to segment the speech inputs, and gray lines indicate the fixed segmentation of 280 ms. For clarity, we use an external alignment tool Forced-Alignment⁹ (Kürzinger et al., 2020) to align the transcription with the speech, where the green area is the speech interval corresponding to the transcription marked by the tool. Note that the alignment provided by external tools is only a rough reference, not necessarily absolutely accurate, and the value in the marked interval represents the probability of alignment.

For the simple case in Figure 12(a), where the speech is short and the correspondence between reference and speech (transcription) is much monotonic, DiSeg can basically accurately segment the speech inputs and achieve high-quality translation. In particular, when both lagging 3 segments, DiSeg achieves much lower latency than Wait-k due to more precise segmentation.

For the hard case in Figure 12(b), where the speech is much longer and contains a long silence, DiSeg can also precisely segment the speech inputs. Besides, there is an obvious word order difference (Zhang and Feng, 2022a) between reference and speech (transcription) in this case, which is more challenging for SimulST (Ma et al., 2019). Since
the fixed segmentation cannot adjust, Wait-k misses translating ‘know’. DiSeg can dynamically adjust the segmentation, and thereby decides to segment and translate ‘weiß’ after receiving ‘know’ in the speech. Owing to precise segmentation, DiSeg can achieve better translation quality under the same latency.

D Numerical Results

D.1 Metrics

For latency, besides Average Lagging (AL) (Ma et al., 2019), we additionally use Consecutive Wait (CW) (Gu et al., 2017), Average Proportion (AP) (Cho and Esipova, 2016) and Differentiable Average Lagging (DAL) (Arivazhagan et al., 2019) to evaluate the latency of DiSeg. Assuming that DiSeg translates $y_t$ at the moment $T(y_t)$, the calculations of latency metrics are as follows.

Consecutive Wait (Gu et al., 2017) CW evaluates the average waiting duration between two adjacent outputs, calculated as:

$$\text{CW} = \frac{\sum_{t=1}^{\text{|y|}} (\mathcal{T}(y_t) - \mathcal{T}(y_{t-1}))}{\sum_{t=1}^{\text{|y|}} \mathbb{1}_{\mathcal{T}(y_t) - \mathcal{T}(y_{t-1}) > 0}},$$

where $\mathbb{1}_{\mathcal{T}(y_t) - \mathcal{T}(y_{t-1}) > 0}$ counts the number of $\mathcal{T}(y_t) - \mathcal{T}(y_{t-1}) > 0$.

Average Proportion (Cho and Esipova, 2016) AP evaluates the average proportion between $\mathcal{T}(y_t)$ and the total duration $T$ of the complete source speech, calculated as:

$$\text{AP} = \frac{1}{|\text{y}|} \sum_{t=1}^{\text{|y|}} \frac{\mathcal{T}(y_t)}{T}.$$  

Average Lagging (Ma et al., 2019, 2020b) AL evaluates the average duration that target outputs lag behind the speech inputs, is calculated as:

$$\text{AL} = \frac{1}{\tau} \sum_{t=1}^{\tau} \mathcal{T}(y_t) - \frac{t - 1}{|\text{y}| / T},$$

where $\tau = \arg\min_t (\mathcal{T}(y_t) - T)$.

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

$$\text{DAL} = \frac{1}{|\text{y}|} \sum_{t=1}^{\text{|y|}} T'(y_t) - \frac{t - 1}{|\text{y}| / T},$$

where

$$\mathcal{T}'(y_t) = \begin{cases} \mathcal{T}(y_t) & t = 1 \\ \max(\mathcal{T}(y_t), \mathcal{T}'(y_{t-1}) + \frac{T}{|\text{y}|}) & t > 1 \end{cases}.$$  

(27)

For translation quality, in addition to SacreBLEU (Post, 2018), we also provide TER (Snover et al., 2006), chrF (Popović, 2015) and chrF++ (Popović, 2017) score of DiSeg.

D.2 Numerical Results

The numerical results of DiSeg with more metrics are reported in Table 4 and Table 5.
### En→De

<table>
<thead>
<tr>
<th>$k$</th>
<th>CW</th>
<th>AP</th>
<th>AL</th>
<th>DAL</th>
<th>BLEU</th>
<th>TER</th>
<th>chrF</th>
<th>chrF++</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>462</td>
<td>0.67</td>
<td>1102</td>
<td>1518</td>
<td>18.85</td>
<td>73.13</td>
<td>44.29</td>
<td>42.31</td>
</tr>
<tr>
<td>3</td>
<td>553</td>
<td>0.76</td>
<td>1514</td>
<td>1967</td>
<td>20.74</td>
<td>69.95</td>
<td>49.34</td>
<td>47.09</td>
</tr>
<tr>
<td>5</td>
<td>666</td>
<td>0.82</td>
<td>1928</td>
<td>2338</td>
<td>22.11</td>
<td>66.90</td>
<td>50.13</td>
<td>47.94</td>
</tr>
<tr>
<td>7</td>
<td>850</td>
<td>0.86</td>
<td>2370</td>
<td>2732</td>
<td>22.98</td>
<td>65.42</td>
<td>50.36</td>
<td>48.23</td>
</tr>
<tr>
<td>9</td>
<td>1084</td>
<td>0.90</td>
<td>2785</td>
<td>3115</td>
<td>23.01</td>
<td>65.48</td>
<td>50.24</td>
<td>48.13</td>
</tr>
<tr>
<td>11</td>
<td>1354</td>
<td>0.92</td>
<td>3168</td>
<td>3464</td>
<td>23.13</td>
<td>65.04</td>
<td>50.42</td>
<td>48.31</td>
</tr>
<tr>
<td>13</td>
<td>1632</td>
<td>0.94</td>
<td>3575</td>
<td>3846</td>
<td>23.05</td>
<td>64.85</td>
<td>50.53</td>
<td>48.41</td>
</tr>
<tr>
<td>15</td>
<td>1935</td>
<td>0.96</td>
<td>3801</td>
<td>4040</td>
<td>23.12</td>
<td>64.92</td>
<td>50.47</td>
<td>48.36</td>
</tr>
</tbody>
</table>

Table 4: Numerical results of DiSeg on MuST-C En→De.

### En→Es

<table>
<thead>
<tr>
<th>$k$</th>
<th>CW</th>
<th>AP</th>
<th>AL</th>
<th>DAL</th>
<th>BLEU</th>
<th>TER</th>
<th>chrF</th>
<th>chrF++</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>530</td>
<td>0.67</td>
<td>1144</td>
<td>1625</td>
<td>22.03</td>
<td>71.34</td>
<td>45.69</td>
<td>43.82</td>
</tr>
<tr>
<td>3</td>
<td>563</td>
<td>0.76</td>
<td>1504</td>
<td>2107</td>
<td>24.49</td>
<td>66.63</td>
<td>53.09</td>
<td>50.85</td>
</tr>
<tr>
<td>5</td>
<td>632</td>
<td>0.81</td>
<td>1810</td>
<td>2364</td>
<td>26.58</td>
<td>63.35</td>
<td>54.55</td>
<td>52.39</td>
</tr>
<tr>
<td>7</td>
<td>788</td>
<td>0.85</td>
<td>2249</td>
<td>2764</td>
<td>27.81</td>
<td>61.87</td>
<td>55.28</td>
<td>53.16</td>
</tr>
<tr>
<td>9</td>
<td>1010</td>
<td>0.89</td>
<td>2694</td>
<td>3164</td>
<td>28.33</td>
<td>60.98</td>
<td>55.51</td>
<td>53.40</td>
</tr>
<tr>
<td>11</td>
<td>1257</td>
<td>0.92</td>
<td>3108</td>
<td>3530</td>
<td>28.59</td>
<td>60.63</td>
<td>55.64</td>
<td>53.55</td>
</tr>
<tr>
<td>13</td>
<td>1534</td>
<td>0.94</td>
<td>3479</td>
<td>3855</td>
<td>28.72</td>
<td>60.49</td>
<td>55.61</td>
<td>53.53</td>
</tr>
<tr>
<td>15</td>
<td>1835</td>
<td>0.96</td>
<td>3819</td>
<td>4160</td>
<td>28.92</td>
<td>60.22</td>
<td>55.80</td>
<td>53.71</td>
</tr>
</tbody>
</table>

Table 5: Numerical results of DiSeg on MuST-C En→Es.
A For every submission:

✓ A1. Did you describe the limitations of your work?
   In Limitations Section.

☐ A2. Did you discuss any potential risks of your work?
   Not applicable. Left blank.

✓ A3. Do the abstract and introduction summarize the paper’s main claims?
   In Section 1.

☒ A4. Have you used AI writing assistants when working on this paper?
   Left blank.

B ✓ Did you use or create scientific artifacts?
   Left blank.

✓ B1. Did you cite the creators of artifacts you used?
   In Section 4 and 5.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
   Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
   Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
   Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
   Not applicable. Left blank.

✓ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
   In Section 4.

C ✓ Did you run computational experiments?
   In Section 4 and 5.

✓ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
   In Section 4.2.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

In Appendix D.

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

Not applicable. Left blank.

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

In Section 4.2.

D. Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

No response.

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?

No response.

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

No response.

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

No response.

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

No response.