INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Impact of Encoding and Segmentation Strategies on End-to-End Simultaneous Speech Translation

Ha Nguyen[1,2]*, Yannick Estève*[2]*, Laurent Besacier*[1,3]

[1]LIG - Université Grenoble Alpes, France
[2]LIA - Avignon Université, France
[3]Naver Labs Europe, France

manh-ha.nguyen@univ-grenoble-alpes.fr, yannick.esteve@univ-avignon.fr, laurent.besacier@naverlabs.com

Abstract

Boosted by the simultaneous translation shared task at IWSLT 2020, promising end-to-end online speech translation approaches were recently proposed. They consist in incrementally encoding a speech input (in a source language) and decoding the corresponding text (in a target language) with the best possible trade-off between latency and translation quality. This paper investigates two key aspects of end-to-end simultaneous speech translation: (a) how to encode efficiently the continuous speech flow, and (b) how to segment the speech flow in order to alternate optimally between reading (R: encoding input) and writing (W: decoding output) operations. We extend our previously proposed end-to-end online decoding strategy and show that while replacing BLSTM by ULSTM encoding degrades performance in offline mode, it actually improves both efficiency and performance in online mode. We also measure the impact of different methods to segment the speech signal (using fixed interval boundaries, oracle word boundaries or randomly set boundaries) and show that our best end-to-end online decoding strategy is surprisingly the one that alternates R/W operations on fixed size blocks on our English-German speech translation setup.

Index Terms: simultaneous speech translation, online sequence-to-sequence models, speech segmentation, efficient speech technologies.

1. Introduction

Online (also known as *simultaneous*) machine translation refers to automatic translation systems which start generating an output hypothesis before the entire input sequence has been consumed [1, 2]. Emerging recently as a challenging task, it has been witnessing several works proposed in text-to-text ($T2T$) translation [3, 4, 5, 6], and in speech-to-text ($S2T$) translation [7, 8, 9, 10], which attempt to deal with the low latency constraint imposed by the task. Following the *wait-k* policy originally proposed for $T2T$ [3] and proven effective when applied to $S2T$ [8, 9], our previous work [10] introduced an adaptive version of *wait-k* which leverages any pre-trained end-to-end offline speech translation model for online speech translation. However, the model proposed in [10] had a speech encoder based on a *Bi-directional* Long Short-Term Memory (BLSTM) [11] which was not efficient in online mode since re-encoding of the full input was needed each time a new speech block was read.

We show in this work that while replacing BLSTM by *Uni-directional* Long Short-Term Memory (ULSTM) encoding degrades performance in offline mode, it actually improves both efficiency and performance in online mode (this observation was also made for online $T2T$ translation by [6]). We also investigate how to segment the speech flow in order to alternate optimally between reading (R: encoding input) and writing (W: decoding output) operations. The contributions of this work are the following:

- Showing that ULSTM speech encoder when using the same (*re-encode*) encoding strategy yields better inference speed and performance in comparison with BLSTM speech encoder,
- Further improving inference speed and performance of ULSTM speech encoder using a new encoding strategy (ULSTM *Overlap-and-Compensate*),
- Analyzing the impact of speech flow segmentation on the BLEU/Latency trade-off, comparing three segmentation methods: fixed interval boundaries, oracle word boundaries or randomly set boundaries.

2. Background on low latency neural speech translation

2.1. Decoding strategies

Real-life applications require translation systems to start emitting output translation partially before the input sequence is made fully available. Such a low latency constraint has been imposing great challenge to neural sequence-to-sequence translation models, despite their state-of-the-art performance on offline translation tasks. Notable efforts have been going into optimizing quality/latency trade-off of the neural online translation systems, including [12] who introduces a waiting policy which alternates READ/WRITE operations. Inspired by [12], [13] designs a static read and write decoding policy, which first reads S input tokens, and alternates between a same number of WRITE and READ operations until the entire source sequence is consumed. In the same spirit, [3] proposes a *wait-k* decoding policy which reads k source tokens at the first step, and then alternates single WRITE/READ operations.

Several works on online automatic speech translation got decent results when adapting *wait-k* policy to their task, including [7, 8, 9, 10]. [9] made an attempt to build an end-to-end online system which first reads k input frames, then alternates between writing one output token or reading the next s input frames. [10] extends this work, modifying their decoding policy to be able to emit more than one (and maximum N) output tokens at a time. The policy of [10] allows them to exploit any pre-trained offline model in an online decoding mode. However, they only experiment with pre-trained models whose speech encoders use BLSTM layers. [6] shows that, in online mode, BLSTM models might be an unnecessarily costly choice, and therefore advocates for using ULSTM models instead (for text translation). In

http://dx.doi.org/10.21437/Interspeech.2021-608

this work, we explore the use of ULSTM models, and make a comparison with their BLSTM counterpart for low latency end-to-end speech translation. We also experiment alternative speech segmentation policies to [10].

2.2. Evaluation metrics

Performance of online translation systems is usually illustrated as a trade-off between translation quality and latency. As in offline translation, BLEU remains the most frequently used metrics for measuring translation quality of online systems. Several metrics have been proposed for latency measurement [12, 3, 14], amongst which Average Lagging (AL) proposed by [3] is a frequent choice. The original AL metric measures the average rate at which the translation system lags behind an ideal *wait-0* translator. [15] argues that this metric has a shortcoming when applied to $S2T$ translation, and proposes an adaptive version which remedies this problem. However, we noticed that this adaptive version is strongly sensitive to the reference's length, which can be arbitrarily long and weakly dependent on the input speech. In some cases, a slight change of the reference length (which might come from a different tokenization method for example) could drastically change the AL value. Furthermore, one should keep in mind that negative values of AL can still occur when the translation system in question gets ahead of the ideal translator (i.e when it predicts output tokens although the already read source frames do not account for them). Despite those shortcomings, we keep using the adaptive AL from [15] in this work in order to measure our improvements of results over those of [10].

3. End-to-end online model

Our previous work [10] reused an attention-based encoder-decoder architecture described in [16]. The speech encoder stacks two VGG-like CNN blocks [17] before five layers of BLSTM. We stack in each VGG block two 2D-convolution layers, followed by a 2D-maxpooling layer. After these two VGG blocks, the shape $(T \times D)$ of an input speech sequence is transformed to $(T/4 \times D/4)$, with T being the length of the input sequence (number of frames), and D being the features' dimension respectively. The decoder is a stack of two 1024-dimensional LSTM layers, and Bahdanau's attention mechanism [18] is used to bridge the encoder and the decoder. In online mode, the BLSTM speech encoder must re-encode from the beginning, from left-to-right and from right-to-left, the input speech sequence every time new input frames are read. In terms of decoding strategy, an adaptive version of *wait-k* is proposed in [10]. This deterministic decoding strategy reads at the first reading operation k (*wait* parameter) first acoustic frames of the input speech features sequence. At each reading operation after this, the system continues consuming fixed intervals of s (*stride* parameter) frames (this reading strategy is also referred in this paper as the fixed interval boundaries segmentation method). A writing operation is put after each reading operation, which writes at maximum N (*write* parameter) output tokens.

ULSTM Re-encode strategy [6] proves that, for $T2T$ online translation, using a ULSTM encoder gives not only better decoding speed but also better BLEU/AL trade-off. We verify if this idea works for speech as well, comparing BLSTM and ULSTM speech encoders in this work. In order to make this comparison, we retrain an offline model similar to the one presented in [10], except that the speech encoder is modified to stacked ULSTM layers instead of BLSTM layers after the VGG-like blocks. In this strategy (presented in figure 1a) we still re-encode the full speech sequence left-to-right every time we read new input frames, but this *ULSTM-Re-encode* approach frees us from computing the BLSTM's right-to-left re-encoding pass, hence being expected to improve decoding speed.

ULSTM Overlap-and-Compensate strategy Moving from BLSTM to ULSTM is a first step towards efficiency but re-encoding the full sequence left-to-right each time speech frames are read is still sub-optimal. To avoid this, we tried to feed chunk by chunk of input frames independently but this solution gave very disappointing results probably because of the quality deterioration of the VGG blocks' output representations due to padding issues near the chunk boundaries (especially in the last several positions of the representations). Therefore, when dealing with ULSTM speech encoders, we propose an *Overlap-and-Compensate* encoding strategy which allows the encoder to read extra frames from the past in order to compensate some discarded positions in the end of the previous output representation of the VGG-like blocks (figure 1b).

Algorithm 1 Overlap-and-Compensate encoding strategy

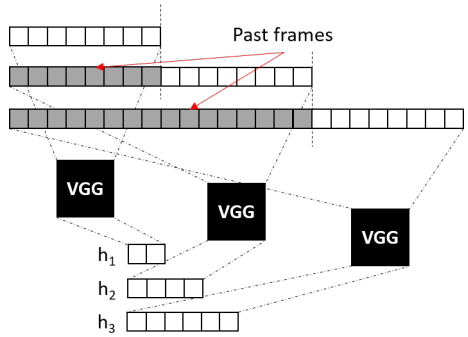
Input: sequence x;
Output: representation h;
Initialization step $t = 1$, wait parameter k, stride parameter s, total number of frames read so far $g = k$, $offset = 0$, $finish\_read = False$, $h_0 = None$,
$overlap = round(k/2)$; *# Overlap half of chunk_size*
while $g < |x|$ **do**
  if $t > 1$ **then**
    $overlap = round(s/2)$;
  end
  if $g >= |x|$ **then**
    $g = |x|$; $overlap = 0$; $finish\_read = True$;
  end
  $x_t = x[offset : g]$; *# A chunk read at time t*
  $h_t = Encode(x_t, overlap, h_{t-1}, finish\_read)$;
  $g+ = s$; $t+ = 1$; $offset = g - overlap$;
end
Fuction `Encode` $(x, overlap, prev\_h, finish\_read)$ **:**
  $num\_discard = round(overlap/4)$;
  $h_{vgg} = VGG(x)$;
  if *not* $finish\_read$ **then**
    # Discard num_discard positions in the end
    $new\_length = |h_{vgg}| - num\_discard$;
    $h_{vgg} = h_{vgg}[0 : new\_length]$;
  end
  return $h_{ULSTM} = ULSTM(h_{vgg}, prev\_h)$;

Algorithm 1 describes the overlap-and-compensate approach applied to the fixed interval segmentation presented in [10]. It introduces another parameter $overlap$, which decides how many past frames the encoder should read at each encoding step (our *re-encode* strategy corresponds to $overlap = 0, offset = 0$). We experiment with $overlap$ corresponding to half of the number of input frames of the current step ($overlap = round(s/2)$).

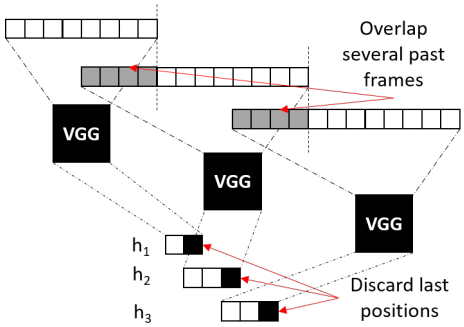
4. Experimental Setup

Data This work focuses on the English-German (EN-DE) language pair. As mentioned in [8], the data used to train our models is a combination of MuST-C EN-DE [19], Europarl EN-DE [20], and How2 [21] synthetic (i.e. the German translation has been automatically generated by a $T2T$ machine translation system), overall more than 750h of translated speech.

Pre-trained models The offline BLSTM model presented in this work was trained for our participation to IWSLT 2020 [8]. It scores 21.38 and 20.54 BLEU on MuST-C tst-COMMON, and



(a) *Re-encode*



(b) *Overlap-and-compensate*

Figure 1: *Different encoding strategies.*

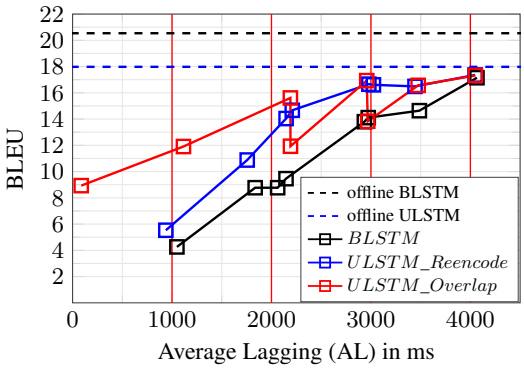
tst-HE, in greedy decoding mode, respectively. We pre-train another offline ULSTM model with exactly the same configuration as the BLSTM model, only replacing BLSTM layers by ULSTM layers. It scores 18.21 and 17.98 BLEU on tst-COMMON, and tst-HE, in greedy decoding mode, respectively.

5. Experiments

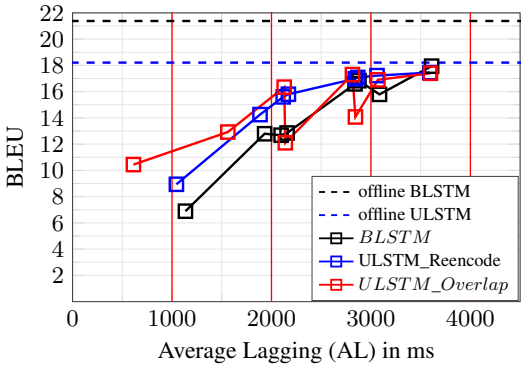
5.1. Impact of encoding strategies

This subsection compares different models using either BLSTM or ULSTM speech encoders with different encoding strategies (*re-encode* versus *overlap-and-compensate*). We use the same segmentation (arbitrarly fixed interval boundaries) presented in [10]. Figure 2 illustrates the BLEU/AL trade-off of BLSTM and ULSTM models with different encoding strategies, evaluated on MuST-C tst-HE and MuST-C tst-COMMON, with different (k, s, N) triplets ($k = [100, 200]$, $s = [10, 20]$, and $N = [1, 2]$). It is noticeable that models with ULSTM speech encoders give consistently better BLEU/AL trade-off than the model with BLSTM speech encoder, on both MuST-C tst-HE and tst-COMMON. Moreover, figure 2 clearly shows that ULSTM *overlap-and-compensate* strategy outperforms ULSTM *re-encode*, especially in low-latency regimes.

We also investigate the actual time spent decoding each sentence of MuST-C tst-HE using different encoding strategies. In order to do this, we exclusively use the same CPU machine to decode the whole test set using either BLSTM, ULSTM *re-encode* or ULSTM *overlap-and-compensate* encoding strategy. Actual time spent decoding each sentence is captured and averaged over the whole test set. To better illustrate the difference between the encoding strategies, in each latency regime, the time spent of BLSTM is set as a speed unit, and the results of ULSTM *re-encode* and ULSTM *overlap-and-compensate* are reported according to this speed unit. We observe that amongst all latency regimes, ULSTM models are much faster than the BLSTM model since they only need to encode the input speech in one direction (from left to right). ULSTM *re-encode* is about twice as fast as BLSTM, scoring 0.53. Remarkably, scoring 0.06, the ULSTM *overlap-and-compensate* is fastest among all encoding strategies (about 17 times faster than the BLSTM, and 9 times faster than ULSTM *re-encode*, respectively). We believe that this huge improvement in terms of computation speed of the ULSTM *overlap-and-compensate* approach is due to the fact that its input chunks are consistently smaller than that of the ULSTM *re-encode* approach.



(a) *MuST-C tst-HE*



(b) *MuST-C tst-COMMON*

Figure 2: *Comparing translation models with BLSTM/ULSTM re-encode/ULSTM Overlap encoding strategies, evaluated on MuST-C tst-HE and tst-COMMON.*

5.2. Impact of speech input segmentation

In this section, we investigate the optimal ways to segment the speech flow in order to alternate between reading (R: encoding input) and writing (W: decoding output) operations: fixed interval boundaries (as presented in [10] and in previous experiments of this paper), oracle word boundaries segmentation, and randomly set boundaries segmentation.

5.2.1. Oracle word boundaries

Questioning whether or not feeding relatively precise word-by-word speech chunks instead of fixed-length chunks [10] would improve the performance, we segment the input audio (phrase level) into words using Montreal Forced Aligner [22]. Their pre-trained English model (from Librispeech [23])[1] is used out of the box. In terms of decoding strategies, we slightly modify

[1] `https://montreal-forced-aligner.readthedocs.io`

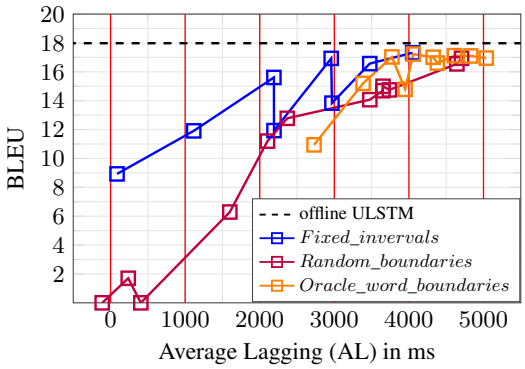


Figure 3: *BLEU/AL trade-off for different speech input segmentation methods, evaluated on MuST-C tst-HE, using ULSTM overlap-and-compensate approach.*

the strategy proposed by [10]:

- k remains the number of frames the encoder should wait before starting writing, serving as an upper bound. At the first decoding step, the encoder reads the first several chunks of frames (matching the words boundaries) until $total\_number\_of\_frames \geq k$. In this work, we experiment with $k = [0, 50, 100, 150, 200]$.
- s is the number of source words (chunks of frames) read at each decoding step after the first step. In this work, we keep $s = 1$ for all our experiments regarding the oracle word boundaries segmentation.
- N remains the maximum number of output tokens (characters) written at each decoding step ($N = [1, 2]$).

5.2.2. *Randomly set boundaries*

The randomly set boundaries segmentation method cuts the audio input into random sized audio chunks. However, to avoid unreasonable fluctuation of the size of each chunk, we set a lower bound (the minimum number of frames) and a higher bound (the maximum number of frames) for each chunk. The number of frames in each chunk is randomly generated within this constraint. We continuously accumulate these random numbers until their sum exceeds the total number of frames in the input sequence. The number of frames in the last chunk is adjusted so that the sum of frames in all chunks is equal to the input sequence's length. In this work, we experiment with $[low\_boundary, high\_boundary] = [5, 10], [5, 20], [5, 50], [5, 100], [10, 50], [10, 100]$. We experiment with $N = [1, 2]$ in this setup.

Algorithm 1 when applied to these two segmentation methods would slightly change: $s = |segment[t]| - |segment[t-1]|$, and $k = |segment[0]|$. Note that as for the oracle word boundaries, $segment[0]$ corresponds to all words read at the first decoding step.

Figure 3 illustrates that the ULSTM *Overlap-and-compensate* encoding strategy performs best with the fixed interval boundaries segmentation. Surprisingly, the oracle word boundaries segmentation does not seem to be beneficial in comparison with the fixed interval boundaries as it almost always takes bigger AL in order to achieve comparable BLEU scores. We suspect that this happens because the average length of each word (37 frames) is much bigger than the stride parameter ($s = 10$ or $s = 20$ frames) that we set for the fixed interval boundaries. Figure 3 also shows that the randomly set boundaries segmentation perform the worst. Their BLEU scores approach 0 (the red dots at the bottom of figure 3) when the segment sizes are too small ($[low\_boundary, high\_boundary] = [5, 10]$).

5.3. Highlighting the most difficult utterances for simultaneous decoding

[24] introduced a metric to measure the lagging difficulty of an utterance: after estimating source-target ((x, y)) alignments (for instance with *fast-align* [25]), they define a non-decreasing function $z^{align}(t)$, denoting the number of source words needed to translate a target word. This function guarantees that at a given decoding position t, $z^{align}(t)$ is larger than or equal to all the source positions aligned with t. Lagging difficulty (LD) is then defined as equation (1) below, with $\tau = argmin_t\{t|z_t = |x|\}$:

$$LD(x, y) = \frac{1}{\tau}\sum_{t=1}^{\tau} z_t^{align} - \frac{|x|}{|y|}(t-1) \quad (1)$$

Based on LD, we extract the 100 most difficult and the 100 easiest sentences according to the metrics, and report the BLEU/AL trade-off for these sets of utterances. Figure 4 shows that LD metrics could be a good tool for highlighting the most difficult utterances for simultaneous decoding since the AL/BLEU curve for the easiest utterances is clearly above the one for the hardest utterances. This suggest the possibility to build specific challenge sets for end-to-end simultaneous speech translation.

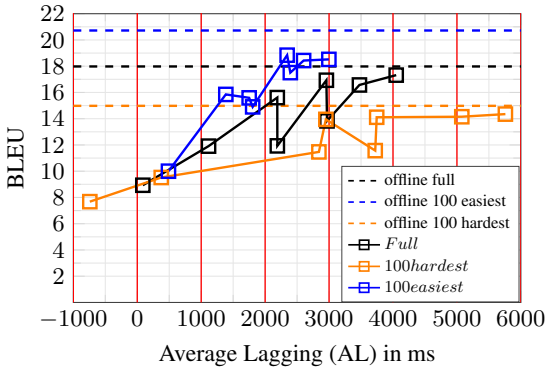


Figure 4: *BLEU/AL trade-off scored on different subsets of MuST-C tst-HE based on Lagging Difficulty (LD).*

6. Conclusions

This paper advocates for using ULSTM instead of BLSTM speech encoder for online translation systems, as it shows that ULSTM outperforms BLSTM in terms of both inference speed and BLEU/AL trade-off. We further improve inference speed and performance of ULSTM speech encoder by proposing a new encoding strategy called ULSTM overlap-and-compensate. Moreover, this work investigates the impact of segmentation on the BLEU/AL trade-off of the ULSTM overlap-and-compensate strategy, and shows that this encoding method works best with equal sized chunks. We also show that difficulty lagging, an indicator of the complexity of the source sentence, might have a great impact on the performance of the online translation systems.

7. Acknowledgements

This work was funded by the French Research Agency (ANR) through the ON-TRAC project under contract number ANR-18-CE23-0021, and was performed using HPC resources from GENCI-IDRIS (Grant 20XX-AD011011365).

8. References

[1] S. Bangalore, V. K. R. Sridhar, P. Kolan, L. Golipour, and A. Jimenez, "Real-time incremental speech-to-speech translation of dialogs," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2012, pp. 437–445.

[2] V. K. R. Sridhar, J. Chen, S. Bangalore, A. Ljolje, and R. Chengalvarayan, "Segmentation strategies for streaming speech translation," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 230–238.

[3] M. Ma, L. Huang, H. Xiong, R. Zheng, K. Liu, B. Zheng, C. Zhang, Z. He, H. Liu, X. Li, H. Wu, and H. Wang, "STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework," in *Proc. of ACL*, 2019. [Online]. Available: https://www.aclweb.org/anthology/P19-1289/

[4] N. Arivazhagan, C. Cherry, W. Macherey, C.-C. Chiu, S. Yavuz, R. Pang, W. Li, and C. Raffel, "Monotonic infinite lookback attention for simultaneous machine translation," in *Proc. of ACL*, 2019. [Online]. Available: https://www.aclweb.org/anthology/P19-1126/

[5] X. Ma, J. Pino, J. Cross, L. Puzon, and J. Gu, "Monotonic multihead attention," in *Proc. of ICLR*, 2020. [Online]. Available: https://openreview.net/forum?id=Hyg96gBKPS

[6] M. Elbayad, L. Besacier, and J. Verbeek, "Efficient Wait-k Models for Simultaneous Machine Translation," in *Interspeech 2020*, Shangai (Virtual Conf), China, Oct. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02962195

[7] J. Niehues, N. Pham, T. Ha, M. Sperber, and A. Waibel, "Low-latency neural speech translation," in *Proc. of INTERSPEECH*, 2018. [Online]. Available: http://arxiv.org/abs/1808.00491

[8] M. Elbayad, H. Nguyen, F. Bougares, N. Tomashenko, A. Caubrière, B. Lecouteux, Y. Estève, and L. Besacier, "ON-TRAC Consortium for End-to-End and Simultaneous Speech Translation Challenge Tasks at IWSLT 2020," in *The International Conference on Spoken Language Translation ACL - 17th IWSLT*, Seattle, WA, United States, Jul. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02895893

[9] H. J. Han, M. A. Zaidi, S. R. Indurthi, N. K. Lakumarapu, B. Lee, and S. Kim, "End-to-end simultaneous translation system for IWSLT2020 using modality agnostic meta-learning," in *Proceedings of the 17th International Conference on Spoken Language Translation*. Online: Association for Computational Linguistics, Jul. 2020, pp. 62–68. [Online]. Available: https://www.aclweb.org/anthology/2020.iwslt-1.5

[10] H. Nguyen, Y. Estève, and L. Besacier, "An empirical study of end-to-end simultaneous speech translation decoding strategies," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

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

[13] F. Dalvi, N. Durrani, H. Sajjad, and S. Vogel, "Incremental decoding and training methods for simultaneous translation in neural machine translation," *arXiv preprint arXiv:1806.03661*, 2018.

[14] C. Cherry and G. Foster, "Thinking slow about latency evaluation for simultaneous machine translation," *arXiv preprint arXiv:1906.00048*, 2019.

[15] C. W. J. G. J. P. Xutai Ma, Mohammad Javad Dousti, "Simuleval: An evaluation toolkit for simultaneous translation," in *Proceedings of the EMNLP*, 2020.

[16] H. Nguyen, N. Tomashenko, M. Z. Boito, A. Caubriere, F. Bougares, M. Rouvier, L. Besacier, and Y. Esteve, "ON-TRAC consortium end-to-end speech translation systems for the IWSLT 2019 shared task," in *Proc. of IWSLT*, 2019.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of ICLR*, 2015.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *Proc. of ICLR*, 2015.

[19] M. A. Di Gangi, R. Cattoni, L. Bentivogli, M. Negri, and M. Turchi, "Must-c: a multilingual speech translation corpus," in *Proc. of NAACL-HLT*, 2019.

[20] J. Iranzo-Sánchez, J. A. Silvestre-Cerdà, J. Jorge, N. Roselló, A. Giménez, A. Sanchis, J. Civera, and A. Juan, "Europarl-ST: A multilingual corpus for speech translation of parliamentary debates," in *Proc. of ICASSP*, 2020.

[21] R. Sanabria, O. Caglayan, S. Palaskar, D. Elliott, L. Barrault, L. Specia, and F. Metze, "How2: a large-scale dataset for multimodal language understanding," in *Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL)*. NeurIPS, 2018. [Online]. Available: http://arxiv.org/abs/1811.00347

[22] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *Interspeech*, vol. 2017, 2017, pp. 498–502.

[23] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP*. IEEE, 2015.

[24] M. Elbayad, M. Ustaszewski, E. Esperança-Rodier, J. V. Manquat, Francis Brunet, and L. Besacier, "Online versus offline nmt quality: An in-depth analysis on english-german and german-english," *COLING*, 2020.

[25] C. Dyer, V. Chahuneau, and N. A. Smith, "A simple, fast, and effective reparameterization of ibm model 2," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 644–648.