Does Simultaneous Speech Translation need Simultaneous Models?

Sara Papi[♣,♢], **Marco Gaido**[♣,♢], **Matteo Negri**[♣], **Marco Turchi**[♠*]
[♣]Fondazione Bruno Kessler
[♢]University of Trento
[♠]Zoom Video Communications
{spapi, mgaido, negri}@fbk.eu, marco.turchi@zoom.us

Abstract

In simultaneous speech translation (SimulST), finding the best trade-off between high output quality and low latency is a challenging task. To meet the latency constraints posed by different application scenarios, multiple dedicated SimulST models are usually trained and maintained, generating high computational costs. In this paper, also motivated by the increased sensitivity towards sustainable AI, we investigate whether a *single model* trained offline can serve both offline and simultaneous applications under different latency regimes without additional training or adaptation. Experiments on en→{de, es} show that, aside from facilitating the adoption of well-established offline architectures and training strategies without affecting latency, the offline solution achieves similar or better quality compared to the standard SimulST training protocol, also being competitive with the state-of-the-art system.

1 Introduction

Many application contexts, such as conferences and lectures, require automatic speech translation (ST) to be performed in real-time. To meet this requirement, Simultaneous ST (SimulST) systems strive not only for high output quality but also for low latency (i.e. the elapsed time between the speaker's utterance of a word and the generation of its translation in the target language). Balancing quality and latency is extremely complex as the two objectives are conflicting: in general, the more a system waits – which implies higher latency – the better it translates thanks to a larger context to rely on.

SimulST models manage the quality-latency trade-off by means of a decision policy: the rule that determines whether a system has to wait for more input or to emit one or more target words. The most popular decision policy is the *wait-k*, a straightforward heuristic that prescribes to wait for a predefined number of words before starting to generate the translation. Initially proposed by Ma et al. (2020b) for simultaneous machine translation (SimulMT), the *wait-k* is now widely adopted in SimulST (Ma et al., 2020b; Ren et al., 2020; Han et al., 2020; Chen et al., 2021; Zeng et al., 2021; Ma et al., 2021) thanks to its simplicity. Apart from *wait-k*, other attempts have been made to develop decision policies learned by the SimulST system itself (Ma et al., 2019; Zaidi et al., 2021; Liu et al., 2021a,b), all resulting in computationally expensive models with limited diffusion.

Regardless of the decision policy, SimulST systems are usually trained simulating the conditions faced at inference time, that is with only a partial input available (Ren et al., 2020; Ma et al., 2020b; Han et al., 2020; Zeng et al., 2021; Ma et al., 2021; Zaidi et al., 2021; Liu et al., 2021a). Since the size of the partial input – and consequently of the context that the SimulST system can exploit to translate – varies according to the latency requirements imposed by real-world applications,[1] several models must be trained and maintained to accommodate different quality-latency trade-offs. This results in high computational costs that contrast with rising awareness on the need to reduce energy consumption (Strubell et al., 2019) towards more sustainable AI (Vinuesa et al., 2020; Schwartz et al., 2020).

So far, the benefits of training systems on partial inputs have been taken for granted and, although works employing models trained in offline mode are documented in literature (Nguyen et al., 2021; Ma et al., 2021), the indispensability of simultaneous training in SimulST has never been demonstrated. With an eye at the burden and environmental impact of training multiple dedicated models for different tasks – offline, simultaneous – and latency

* Work done when working at FBK.

[1]For instance, the IWSLT SimulST shared task defines three latency regimes (Anastasopoulos et al., 2021) – $1s$, $2s$, and $4s$ – and limits of acceptability have been set between $2s$ and $6s$ for the *ear-voice span* depending on different conditions and language pairs (Yagi, 2000; Chmiel et al., 2017).

regimes, in this work we address the following question: *Does simultaneous speech translation actually need models trained in simultaneous mode?* To this end, we experiment with a single, easy-to-maintain offline model, which can effectively serve both the simultaneous and offline tasks. Specifically, we explore the application of the widely adopted *wait-k* policy to the offline-trained ST system only at inference time, bypassing any additional training neither to adapt the model to the simultaneous scenario nor to accommodate different latency requirements.[2] Through experiments on two language directions (en→{de, es}), having respectively different and similar word ordering with respect to the source, we show that:

- In terms of sustainability, offline training yields considerable reductions – by a factor of 9 in our evaluation setting – in carbon emission and electricity consumption (Section 4).
- The offline-trained model outperforms or is on par with those trained in simultaneous within the *wait-k* policy framework (Section 5);
- Recent advancements in offline architectures and training strategies further improve output quality without affecting latency (Section 6);
- The effectiveness of offline training also emerges in comparison with the state of the art in SimulST (Liu et al., 2021b): except for the lowest latency regime, our system is superior in the $2s$-$4s$ latency interval (ear-voice span) with gains up to 4.0 BLEU (Section 7).

2 Background

2.1 *wait-k*

The *wait-k* policy requires to wait for a predefined number of words before starting to translate. For instance, a system using a wait-3 policy generates the 1st target word when it receives the 4th source word, the 2nd target word when it receives the 5th source word, and so on. The number of words to wait is controlled by the k parameter. SimulST systems based on the *wait-k* policy are usually trained considering the same k used for testing (Ren et al., 2020; Ma et al., 2020b; Zeng et al., 2021) while, in theory, its value can be different between the training and testing phases. A parameter k_{train} can indeed be used to mask words at training time, while a parameter k_{test} can be used to directly control the latency of the system at inference time according to the requirements posed by the target application scenario.

Since many values of k_{train} can be used to train the SimulST systems, even for identical values of k_{test}, the standard approach involves performing several trainings to obtain the best translation quality while satisfying different latency requirements. In SimulMT, Elbayad et al. (2020) tried to avoid this large number of experiments by exposing the model to different values of k_{train} sampled at each iteration. Surprisingly, they achieve the best performance on several k_{test} using a single value of k for training ($k_{train} = 7$). However, it is not clear if such a rule applies to SimulST, leaving the problem of performing a large number of trainings still unsolved.

2.2 Word detection for *wait-k* in SimulST

Since SimulMT operates on a stream of words, applying the *wait-k* is straightforward because the number of received words is explicit in the input. Conversely, its application to SimulST is complicated by the fact that the input is an audio stream and the number of received words has to be inferred by means of a so-called *word detection* strategy.

Two main categories of word detection strategies are currently employed by the community: fixed (Ma et al., 2020b), and adaptive (Ma et al., 2020b; Ren et al., 2020; Zeng et al., 2021; Chen et al., 2021). The fixed strategy is the easiest approach, as it assumes that a fixed amount of time is required to pronounce every word disregarding the information actually contained in the audio. In contrast, adaptive word detection determines the number of uttered words by looking at the content of the audio. This can be done either by means of an Automatic Speech Recognition (ASR) decoder (Chen et al., 2021),[3] or by means of a Connectionist Temporal Classification (Graves et al., 2006) – CTC – module (Ren et al., 2020; Zeng et al., 2021), every time a speech chunk is received by the system.

In its simplicity, the fixed strategy does not consider various aspects of the input speech, such as different speech rates, duration, pauses, and silences. For instance, if there are no words in the speech (e.g. in the case of pauses or silences), the

[2] Code available at https://github.com/hlt-mt/FBK-fairseq.

[3] This solution involves the use of two separate synchronized decoders (one for simultaneous ASR and one for ST) and will not be analyzed in this work due to the higher computational costs of training a double decoder architecture.

fixed strategy forces the system to output something even if it cannot rely on sufficient context. In the opposite case, in which more than one word is pronounced in a speech chunk, the fixed strategy forces the emission of only one word, consequently accumulating a delay. By trying to guess the actual number of words contained in a speech chunk, the adaptive strategy is in principle more faithful to these audio phenomena. However, conflicting results are reported in literature, some in support of the adaptive strategy (Zeng et al., 2021) while others showing no advantage from its application (Ma et al., 2020b).

3 Do we need Simultaneous training?

While at training time the SimulST system has the entire audio available, at inference time it receives a partial, incremental input. This mismatch between offline training and simultaneous testing makes the system vulnerable to exposure bias (Ranzato et al., 2016). To mitigate this potential problem, SimulST models are trained under simulated simultaneous conditions. On an attentive model, this simultaneous training is realized by masking future audio frames when computing the encoder-decoder attention. For a *wait-k* SimulST system, the choice of the audio frames to be masked depends on two factors: the value of k_{train} and the word detection strategy. The k_{train} value determines the number of source words to mask (e.g., in the case of wait-3, the first target word is generated by looking at the first three source words and so on). The word detection strategy identifies the source words from the audio by detecting the number of frames each one corresponds to. Thus, the encoder-decoder attention is computed by limiting each target word to only attend to the audio frames that correspond to the previous k_{train} source words identified by the word detection strategy. As a result, testing different word detection strategies requires training several systems, which in turn are trained with different values of k_{train} to obtain different latencies.

In this paper, we question the need of all these experiments by investigating whether the simultaneous training of the ST systems is indispensable to obtain a good quality-latency trade-off. Within the framework of the *wait-k* policy, we explore the ability to translate in real-time of an offline-trained system that is neither trained nor adapted to the simultaneous scenario. To obtain a simultaneous prediction from the offline system, we add a *pre-decision module* after the encoder at inference time. Its role is to incorporate the logic of the word detection strategy to decide whether to wait or to emit words when a new speech chunk is received, according to the selected k_{test}. In particular, it takes as input the encoder states representing the received audio chunk and applies the word detection strategy (either fixed or adaptive) to obtain the number of source words present in the input. If this number is equal or exceeds k_{test}, the module activates the decoding part of the model and a word is emitted, otherwise it keeps reading the source speech.

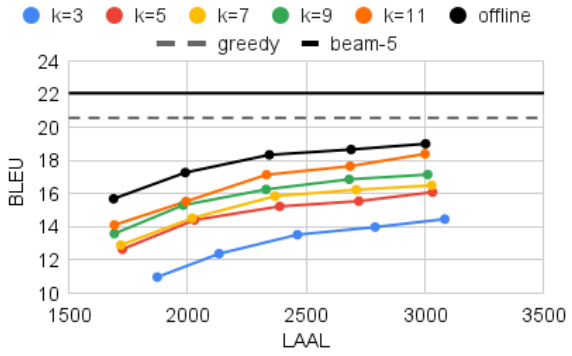
Since the offline system is not trained for the simultaneous task, the choice of k_{test} and word detection strategy are not constrained to those used during training as in the native SimulST case. Indeed, an offline model is trained by always attending to the entire source input. Different from the simultaneous training mode, the encoder-decoder attention is computed without masking, that is by considering past, current, and future information. Although this avoids multiple training for each k_{train} and word detection strategy, it also exposes the model to operate in conditions different from its training setup, as it is not used to receive partial inputs. To check if the exposure bias given by this mismatch in training and testing conditions constitutes a real limitation, we conduct a systematic analysis of the quality-latency performance of the offline-trained system in the simultaneous scenario. To this aim, we compare the offline-trained system with the same model trained in simultaneous mode by varying the value of k_{train} and the word detection strategy.

4 Experimental Settings

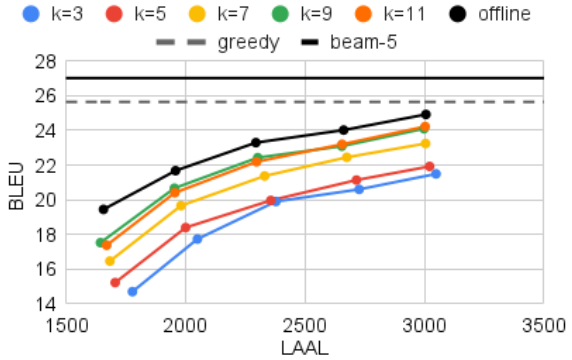
We perform all our experiments on the en→{de, es} sections of the MuST-C dataset (Cattoni et al., 2021). All the results presented are given on the corpus test set (tst-COMMON). We use the Transformer architecture (Vaswani et al., 2017) with the integration of the CTC in the encoder (Liu et al., 2020; Gaido et al., 2021), which is used to realize the adaptive word detection strategy. The hyperparameters, training and inference details are presented in Appendix A.1.

For the evaluation, we adopt BLEU[4] (Post, 2018) for quality, and Length Adaptive Average Lagging (Papi et al., 2022) – or LAAL – for latency, which is the modified version of the popular Average Lag-

[4]BLEU+case.mixed+smooth.exp+tok.13a+version.1.5.1

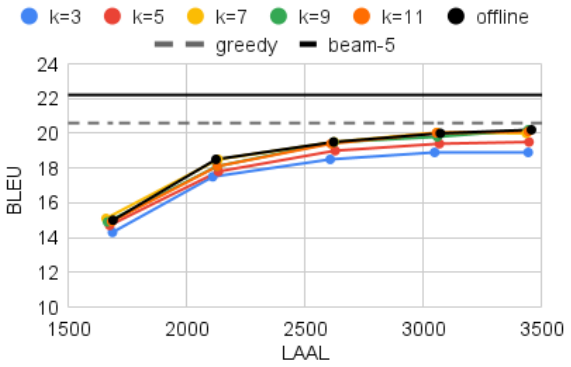


(a) English → German

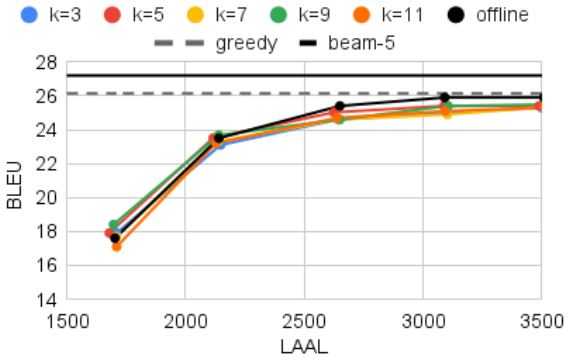


(b) English → Spanish

Figure 1: LAAL-BLEU curves of *wait-k* with fixed word detection strategy.



(a) English → German



(b) English → Spanish

Figure 2: LAAL-BLEU curves of *wait-k* with adaptive word detection strategy.

ging for speech (Ma et al., 2020b) that correctly evaluates both shorter and longer predictions with respect to the reference. We report the simultaneous results in LAAL-BLEU graphs where each curve corresponds to a system trained using a different value of k_{train} and each point to a different k_{test}. The set of k values used for both training the simultaneous model and testing all the models is $k = \{3, 5, 7, 9, 11\}$. We also report the results of the offline generation using the greedy search and the beam search with the $beam\_size = 5$ commonly used in offline ST.

Carbon Footprint. Each training contributed an estimate of 70.3 kg of CO_{2eq} to the atmosphere and used 184.7 kWh of electricity. This assumes 116 hours of runtime, a carbon intensity of 380.539g CO_{2eq} per kWh, 4 NVIDIA Tesla K80 GPUs (utilization 93%), and an Intel Xeon CPU E5-2683 v4 (utilization 100%).[5] This means that training a single offline model instead of a model for each value of k_{train} (in our case, 5 models) and for each word detection strategy (in our case, 2 strategies) allows us to save $5 \cdot 2 - 1 = 9$ experiments, amounting to 632.7 kg of CO_{2eq} and 1662.3 kWh of electricity for each language.

5 Results

Fixed Word Detection. The results of the *wait-k* models with fixed word detection are shown in Figure 1. The LAAL-BLEU curves indicate that the latency of all the systems lies between $1700ms$ and $3000ms$, staying in a medium-high latency regime[6] for both language pairs. Translation quality is lower for en-de, for which it ranges from 11 to 19 BLEU, while for en-es it ranges from 14 to 25 BLEU. The difference in performance between the two language pairs is coherent with the results of the offline generations (both greedy and beam-5) and justified by the different level of difficulty when translating into the two target languages (having respectively similar and different

[5]The social cost of carbon uses models from (Ricke et al., 2018) and carbon emissions information was estimated using the *experiment-impact-tracker* (Henderson et al., 2020).

[6]Henceforth referring to (Anastasopoulos et al., 2021), we consider three latency regimes depending on the delay d between the time in which the speech is heard and the output translation is received. These are: *low* when d<$1000ms$, *medium* when 1000<d<$2000ms$, and *high* when d>$2000ms$.

word ordering with respect to English). The curves of the simultaneous-trained systems also show a tendency: if k_{train} increases, both the quality and latency improve (e.g. on en-de, the k=11 curve lies higher – indicating better quality – and more leftward – lower latency – than the others). Interestingly, the offline-trained models (in solid black) outperform the systems trained in simultaneous at every latency regime, with gains from 1 to 7 BLEU for en-de and from 1 to 6 BLEU for en-es. This indicates that, to achieve the best performance and independently from the k_{test} used, the offline-trained model represents the best choice, at least for the fixed strategy.

Adaptive Word Detection. The results of the *wait-k* models with adaptive word detection are shown in Figure 2. The systems latency lies between $1700ms$ and $3500ms$ and, as with the fixed strategy, the quality is higher for en-es (from 15 to 26 BLEU) than for en-de (from 14 to 20 BLEU). Looking at Figures 1 and 2, we observe that the overall translation quality yielded by the adaptive strategy is higher compared to that of the fixed one. Moreover, the fixed strategy curves are far from being comparable with their offline greedy values (dashed lines), while the adaptive strategy curves almost reach them at higher latency. However, the models with fixed word detection perform better at lower latency, with a gain of 1 BLEU for en-de and 2 BLEU for en-es. In light of these results, there is not a clear winner between the two word detection strategies. From Figure 2, we also notice that the adaptive curves are very close to each other, in contrast with the fixed case. This phenomenon indicates that, in the case of the adaptive strategy, changing k_{train} does not significantly influence the model performance. This suggests that the offline-trained model (comparable to a model trained with $k_{train} = \infty$) should be on par with the simultaneous-trained ones, a consideration corroborated by the trend of the offline-trained system curves (in solid black) that are always above or on par with those of the simultaneous-trained systems.

All in all, we can conclude that, when using the *wait-k* policy, **the offline-trained model achieves similar or even better results compared to the same models trained in simultaneous mode**. Based on this finding, in the next section we explore the actual potential of offline training for SimulST by adopting the most promising offline architectures and training techniques to improve the quality-latency balancing of our systems.

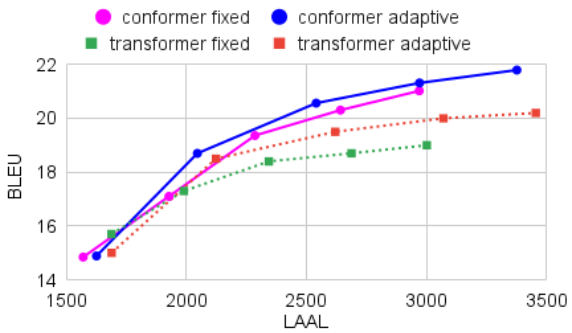
6 Leveraging Offline Solutions

Offline training brings considerable advantages in terms of reducing the computational costs of SimulST technology. First, only one model can be trained and maintained to serve both offline and simultaneous tasks without performance degradation. Second, contrary to the simultaneous-training mode, the choice of the word detection strategy at run-time does not depend on the strategy used during training. Rather, it can be made according to the specific use case, making the offline-trained model more flexible. This also means that other decision policies can be applied to the offline-trained system without the need to re-train it from scratch.

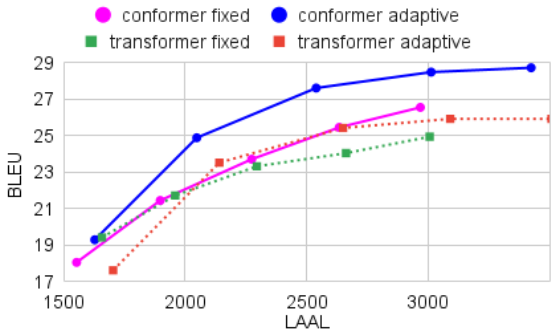
Using a single offline-trained model not only speeds up its development but also opens up the possibility to directly adopt powerful offline architectures and techniques without performing any additional training nor adaptation to the simultaneous scenario. In the following, we test this hypothesis to find out whether recent architectural improvements (Section 6.1) and data augmentation techniques (Section 6.2) designed for offline ST also have a positive impact in SimulST.

In recent years, many architectures have been proposed to address the offline ST task (Wang et al., 2020; Inaguma et al., 2020; Le et al., 2020; Papi et al., 2021). Among them, the Conformer (Gulati et al., 2020) has recently shown impressive results both in speech recognition, for which it was initially proposed, and in speech translation (Inaguma et al., 2021). The main aspects characterizing this encoder-decoder architecture are related to the encoder part. Inspired by the Macaron-Net (Lu et al., 2019), the Conformer encoder is built with a sandwich structure and integrates the relative sinusoidal positional encoding scheme (Dai et al., 2019).

Given the promising results it achieved in the offline scenario, we choose to test if this architecture also brings quality and latency gains in SimulST. Since we found in Section 3 that fixed and adaptive word detection strategies have their own use cases (their best results are observed at different latency regimes, respectively low for fixed and medium-high for adaptive), we compare Conformer- and Transformer-based architectures using both strategies. For the offline training of Conformer, we follow the same procedure used for Transformer. Details about the model hyper-parameters are pre-

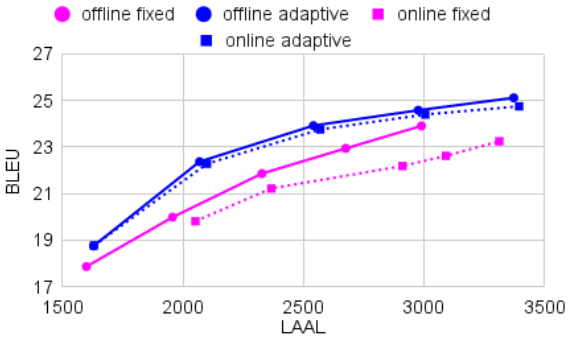


(a) English → German

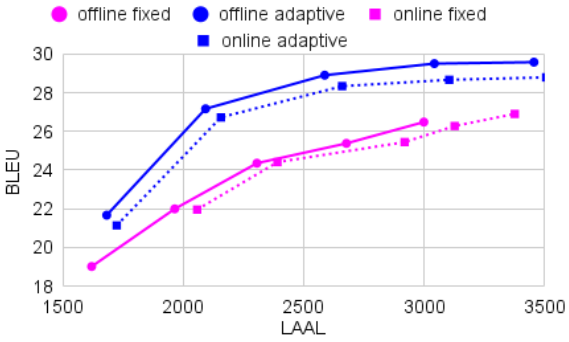


(b) English → Spanish

Figure 3: LAAL-BLEU curves of the Transformer- and Conformer-based architectures.



(a) English → German



(b) English → Spanish

Figure 4: LAAL-BLEU curves of offline- and simultaneous-trained Conformer models with sequence-level KD.

sented in Appendix A.2.

6.1 Scaling Architecture

The offline results of both architectures are presented in Table 1, while their simultaneous curves are shown in Figure 3.

Model	En-De		En-Es	
	greedy	beam-5	greedy	beam-5
Transformer	20.6	22.2	26.1	27.2
Conformer	23.3	24.8	28.5	29.6

Table 1: BLEU results of the offline generation.

As previously noticed by Inaguma et al. (2021), Conformer outperforms Transformer in offline generation. The improvements, of at least 2.4 BLEU points, are valid both for greedy and beam search. From Figure 3, we can see that Conformer outperforms Transformer also in the simultaneous setting. This holds both for fixed and adaptive word detection, with larger BLEU gains at higher latency regimes. As far as word detection strategies are concerned, we also notice a similar trend between Conformer and Transformer: the fixed one performs better or on par at lower latency while being outperformed by the adaptive one when the latency increases.

In light of the better results obtained by Conformer, we conclude that **improving the architecture of the offline system also has a positive impact on its simultaneous performance**, enhancing translation quality without affecting latency.

6.2 Scaling Data

Data augmentation is a common practice used to improve systems performance. One approach to data augmentation is to apply knowledge distillation (KD), which was introduced to transfer knowledge from big to small models (Hinton et al., 2015). Among the possible methods, sequence-level KD (Kim and Rush, 2016) is one of the most popular ones in ST thanks to its application simplicity and the consistent improvements observed (Potapczyk and Przybysz, 2020; Xu et al., 2021; Gaido et al., 2022a). Sequence-level KD consists of replacing the target references of a given parallel training corpus with the predicted sequences generated by a teacher model (usually, an MT model), from which we want to distill the knowledge to a student model.

To investigate the effects of such a knowledge transfer on quality and latency, we apply sequence-level KD to our offline-trained SimulST system.

To this end, we translate the transcripts present in the en→{de, es} sections of MuST-C with an MT model (more details are provided in Appendix A.3) and we substitute the gold translations with the MT-generated ones to build new data. As in (Liu et al., 2021b), to train the models we use both gold and synthetic data by concatenating them. Since the performance of the Conformer model scales with data (Gaido et al., 2022b) and is better compared to that of Transformer (Section 6.1), we adopt the Conformer for the following study. We extend our analysis to the simultaneous-trained systems to verify if the offline-trained one continues to perform at least on par with them and we report the best simultaneous-trained system curve for each word detection strategy.

The effects of the additional KD data are shown in Figure 4. Compared to Figure 3, we notice a performance improvement that comes without sacrificing latency. On en-de, the quality of the offline-trained Conformer with KD ranges from 18 to 25 BLEU, against the previous 15 to 22 BLEU. On en-es, it ranges from 19 to 30 BLEU, against the previous 18 to 29 BLEU. Moreover, the offline-trained system (solid curves) is still better or at least comparable with the simultaneous-trained ones (dotted curves) for both language pairs. From Figure 4, we also notice that adaptive word detection (blue curves) shows overall better results compared to the fixed one (pink curves), even at lower latency. This suggests that comparing the two strategies by using models with higher translation quality shows the superiority of adaptive word detection at any latency regime.

In light of these results, we conclude that **data augmentation improves the offline-trained system quality without affecting latency**. To better assess these performance gains in the simultaneous framework, in the next section we present a detailed comparison of our offline-trained Conformer with the state-of-the-art SimulST architecture.

7 Comparison with the state of the art

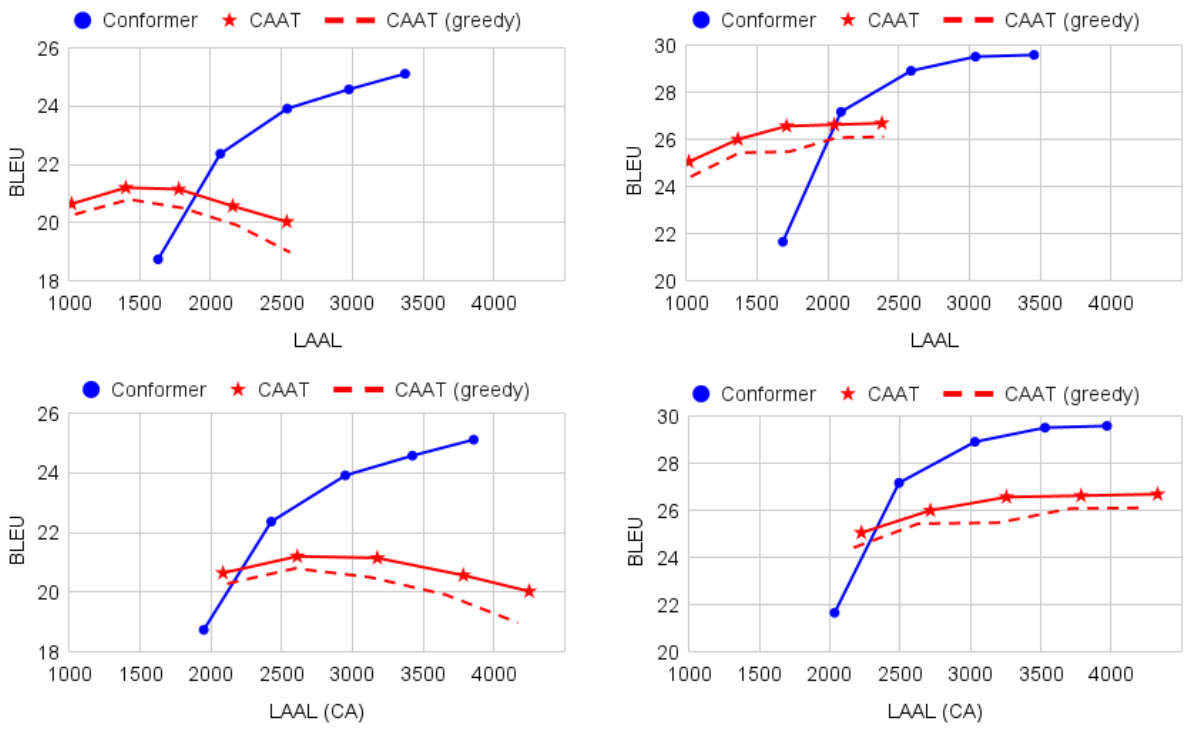
So far, we discovered that scaling to better performing architectures and more data further improves the simultaneous results of offline-trained models. But how good is their performance compared to the state of the art in SimulST? To answer this question, we compare our best system, the offline-trained Conformer with adaptive word detection, with the Cross Attention Augmented Transducer (Liu et al., 2021b) – CAAT – used by the winning submissions at IWSLT 2021 (Anastasopoulos et al., 2021) and 2022 (Anastasopoulos et al., 2022). Inspired by the Recurrent Neural Network Transducer by Graves (2012), CAAT is made of three Transformer stacks: the encoder, the predictor, and the joiner. These three elements are jointly trained in simultaneous to optimize the quality of the translations while keeping latency under control.

For training and testing the CAAT architecture, we use the code published by the authors and adopt the same hyper-parameters of their paper. As the performance of the CAAT model is sensitive to sequence-level KD (Liu et al. 2021b show a 2 BLEU degradation without it), we compare it with the offline-trained Conformer model using the same data settings – see Section 6.2. We report the CAAT results obtained by adopting both the greedy search used in our SimulST settings and the beam search used by Liu et al. (2021b). As suggested by Ma et al. (2020b), we also compute the Computational Aware (CA) version of the LAAL metric (LAAL_{CA}), which is defined as the time elapsed from the beginning of the generation process to the prediction of the partial target.[7] Since LAAL_{CA} represents the real wall-clock elapsed time experienced by the user, it gives a more reliable evaluation of the SimulST performance in a real-time scenario. For the sake of completeness, we also report the results of Average Lagging (Ma et al., 2020a) in Appendix C.

We present the comparison in Figure 5. From the LAAL-BLEU curves, we see that, at low latency regime, the CAAT model (in solid red) outperforms our offline-trained Conformer model (in solid blue) by 2 BLEU on en-de and 4 BLEU on en-es. However, moving to medium-high latency regime, the Conformer significantly outperforms CAAT, reaching gains of 4 BLEU on en-de and 2 BLEU on en-es. We can also notice a degradation of the CAAT en-de translation quality that is caused by an under-generation problem at higher latency, for which we give details in Appendix B.

When it comes to LAAL_{CA}-BLEU, the scenario changes, bringing CAAT curves much closer to those of Conformer. The state of the art still out-

[7] Given that LAAL_{CA} depends on the computation time, we perform all the generations on one NVIDIA Tesla K-80 GPU and provide the results by averaging over 3 runs. However, we notice a very small variance among the runs (in the order of $10ms$), suggesting that averaging is not necessary to provide sound results.



(a) English → German

(b) English → Spanish

Figure 5: LAAL/LAAL$_{CA}$-BLEU curves of our offline-trained Conformer and state of the art (CAAT) models.

performs the Conformer at lower latency but in this case, waiting about $100/200ms$ more, the Conformer performance starts to improve consistently.

Comparing the LAAL- and LAAL$_{CA}$-BLEU curves, we see that our offline-trained system is more coherent between computational and non-computational aware metrics: while Conformer has a computational overhead of $400/500ms$, CAAT requires $1400/1500ms$ more than its ideal LAAL. The CAAT greedy curves (dotted red) show only a little improvement in latency compared to the beam search (solid red), suggesting that its higher computational cost does not depend on the generation strategy but on other factors like its complex and more computationally expensive architecture.

All in all, we can say that, **compared to the state of the art in SimulST, the lower performance of our offline-trained Conformer at low latency regime is balanced by consistently higher BLEU scores at medium and high latency.**

8 Conclusions

To reduce the potentially large amount of experiments usually performed to build SimulST models, we explored the use of a single offline-trained model to serve both the offline and simultaneous tasks. Through comparison with native SimulST systems, we showed that our offline-trained model can be successfully used in real-time, achieving comparable or even better results. To further enhance its performance, we investigated the adoption of consolidated techniques and emerging architectures from offline research, showing consistent improvements also in the simultaneous scenario. The benefits of offline training indicate the potential of applying this method without the need for any additional training or adaptation. Besides facilitating system deployment, another important advantage of building and reusing one single model to rule both tasks is the drastic reduction of the carbon footprint of ST training (by a factor of 9 in our evaluation setting). This represents an important step in response to rising concerns about the AI energy consumption and environmental impact toward more sustainable development.

As regards SimulST evaluation, the differences between results computed with non-computationally and computationally aware latency metrics suggest that including computational time in the measurements heavily influences the outcomes of system comparisons. In our particular case, the differences in latency between the offline-trained models and the state of the art ob-

served in terms of the non-computationally aware LAAL metric become smaller when considering its computational aware version. Although lower latency is theoretically reached by the state of the art CAAT model, this comes at the cost of a more complex and computationally expensive architecture that shows its limitations at inference time. We therefore invite the SimulST community to use computationally aware metrics for more sound evaluations, referring to ideal metrics only in the absence of similar testing assets, as machines with comparable computational power.

9 Limitations

Although it relies on a simpler architecture and generation strategy compared to the state-of-the-art in SimulST, our offline-trained model exhibits a high translation quality in real-time, which allows it to achieve better results at medium and high latency regimes. However, a performance gap of 2-3 BLEU points is still observed at low latency regime. This can be attributed to the use of a simple policy such as *wait-k*. Being the most popular and widely adopted one, we chose to focus on this policy to conduct our analysis. Notwithstanding, investigating better performing solutions to boost performance at low latency and close the gap is still necessary, and definitely among our future work priorities.

Also, the experiments presented in the paper are limited to two target languages, which were selected as representatives of those having similar and different word ordering with respect to the English source speech. Although this choice allowed us to reliably test our hypotheses in diverse conditions, verifying our findings on a wider set of languages is another natural evolution of this research.

Acknowledgments

This work has been supported by the project Smarter Interpreting[8] financed by CDTI Neotec funds, by the ISCRAB project DireSTI granted by CINECA, and by the project "AI@TN" funded by the Autonomous Province of Trento.

References

Antonios Anastasopoulos, Loïc Barrault, Luisa Bentivogli, Marcely Zanon Boito, Ondřej Bojar, Roldano Cattoni, Anna Currey, Georgiana Dinu, Kevin Duh, Maha Elbayad, Clara Emmanuel, Yannick Estève, Marcello Federico, Christian Federmann, Souhir Gahbiche, Hongyu Gong, Roman Grundkiewicz, Barry Haddow, Benjamin Hsu, Dávid Javorský, Věra Kloudová, Surafel Lakew, Xutai Ma, Prashant Mathur, Paul McNamee, Kenton Murray, Maria Nădejde, Satoshi Nakamura, Matteo Negri, Jan Niehues, Xing Niu, John Ortega, Juan Pino, Elizabeth Salesky, Jiatong Shi, Matthias Sperber, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Yogesh Virkar, Alexander Waibel, Changhan Wang, and Shinji Watanabe. 2022. Findings of the IWSLT 2022 evaluation campaign. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 98–157, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Antonios Anastasopoulos, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alex Waibel, Changhan Wang, and Matthew Wiesner. 2021. Findings of the IWSLT 2021 Evaluation Campaign. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, Online.

Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2021. Must-c: A multilingual corpus for end-to-end speech translation. *Computer Speech & Language*, 66:101155.

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.

A. Chmiel, A. Szarkowska, Danijel Korzinek, Agnieszka Lijewska, Łukasz Dutka, Łukasz Brocki, and K. Marasek. 2017. Ear–voice span and pauses in intra- and interlingual respeaking: An exploratory study into temporal aspects of the respeaking process. *Applied Psycholinguistics*, 38:1201 – 1227.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy.

Mattia A. Di Gangi, Marco Gaido, Matteo Negri, and Marco Turchi. 2020. On target segmentation for direct speech translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 137–150, Virtual. Association for Machine Translation in the Americas.

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

[8] https://smarter-interpreting.eu/

chine Translation. In *Proc. Interspeech 2020*, pages 1461–1465.

Marco Gaido, Mauro Cettolo, Matteo Negri, and Marco Turchi. 2021. CTC-based compression for direct speech translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 690–696, Online. Association for Computational Linguistics.

Marco Gaido, Matteo Negri, and Marco Turchi. 2022a. Direct speech-to-text translation models as students of text-to-text models. *IJCoL. Italian Journal of Computational Linguistics*, 8(8-1).

Marco Gaido, Sara Papi, Dennis Fucci, Giuseppe Fiameni, Matteo Negri, and Marco Turchi. 2022b. Efficient yet competitive speech translation: FBK@IWSLT2022. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 177–189, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

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.

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.

Hou Jeung Han, Mohd Abbas Zaidi, Sathish Reddy Indurthi, Nikhil Kumar Lakumarapu, Beomseok Lee, and Sangha Kim. 2020. End-to-end simultaneous translation system for IWSLT2020 using modality agnostic meta-learning. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 62–68, Online. Association for Computational Linguistics.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. In *Proc. of NIPS Deep Learning and Representation Learning Workshop*, Montréal, Canada.

Hirofumi Inaguma, Yosuke Higuchi, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. 2021. Non-autoregressive end-to-end speech translation with parallel autoregressive rescoring. *arXiv preprint arXiv:2109.04411*.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe. 2020. ESPnet-ST: All-in-one speech translation toolkit. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 302–311, Online.

Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. In *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

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.

Hang Le, Juan Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. 2020. Dual-decoder transformer for joint automatic speech recognition and multilingual speech translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3520–3533, Barcelona, Spain (Online).

Dan Liu, Mengge Du, Xiaoxi Li, Yuchen Hu, and Lirong Dai. 2021a. The USTC-NELSLIP systems for simultaneous speech translation task at IWSLT 2021. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 30–38, Bangkok, Thailand (online). Association for Computational Linguistics.

Dan Liu, Mengge Du, Xiaoxi Li, Ya Li, and Enhong Chen. 2021b. Cross attention augmented transducer networks for simultaneous translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 39–55, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yuchen Liu, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2020. Bridging the modality gap for speech-to-text translation. *arXiv preprint arXiv:2010.14920*.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Un-

derstanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*.

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, James Cross, Liezl Puzon, and Jiatao Gu. 2019. Monotonic multihead attention. *arXiv preprint arXiv:1909.12406*.

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, Yongqiang Wang, Mohammad Javad Dousti, Philipp Koehn, and Juan Pino. 2021. Streaming simultaneous speech translation with augmented memory transformer. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7523–7527. IEEE.

Ha Nguyen, Yannick Estève, and Laurent Besacier. 2021. 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)*, pages 7528–7532. IEEE.

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 NAACL-HLT 2019: Demonstrations*.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2021. Speechformer: Reducing information loss in direct speech translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1698–1706, Online and Punta Cana, Dominican Republic.

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.

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, Belgium, Brussels. Association for Computational Linguistics.

Tomasz Potapczyk and Pawel Przybysz. 2020. SR-POL's system for the IWSLT 2020 end-to-end speech translation task. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 89–94, Online.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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.

Katharine Ricke, Laurent Drouet, Ken Caldeira, and Massimo Tavoni. 2018. Country-level social cost of carbon. *Nature Climate Change*, 8(10):895.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):54–63.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Jörg Tiedemann. 2016. OPUS – parallel corpora for everyone. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation: Projects/Products*, Riga, Latvia.

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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ricardo Vinuesa, Hossein Azizpour, Iolanda Leite, Madeline Balaam, Virginia Dignum, Sami Domisch, Anna Felländer, Simone Daniela Langhans, Max Tegmark, and Francesco Fuso Nerini. 2020. The role of artificial intelligence in achieving the sustainable development goals. *Nature communications*, 11(1):1–10.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq S2T: Fast speech-to-text modeling with fairseq. 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: System Demonstrations, pages 33–39, Suzhou, China.

Chen Xu, Xiaoqian Liu, Xiaowen Liu, Tiger Wang, Canan Huang, Tong Xiao, and Jingbo Zhu. 2021. The NiuTrans end-to-end speech translation system for IWSLT 2021 offline task. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 92–99, Bangkok, Thailand (online).

Sane Yagi. 2000. Studying style in simultaneous interpretation. *Meta*, 45(3):520–547.

Mohd Abbas Zaidi, Beomseok Lee, Nikhil Kumar Lakumarapu, Sangha Kim, and Chanwoo Kim. 2021. Decision attentive regularization to improve simultaneous speech translation systems. *arXiv preprint arXiv:2110.15729*.

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.

A Models Architecture

A.1 Transformer

The models used in Section 5 are based on a 12 encoder and 6 decoder layers of Transformer (Vaswani et al., 2017) architecture. The embedding dimension is set to 256, the number of attention heads to 4 and the feed-forward embedding dimension to 2048, both in the encoder and in the decoder. The number of parameters is $\sim$ 32.4M We use Fairseq (Ott et al., 2019) library for all the trainings. The *wait-k* with fixed word detection strategy was already present in the Fairseq library, while we implemented the adaptive one.

We use the hyper-parameters of (Ma et al., 2020b) for all the trainings of the Transformer-based model. We use a unigram SentencePiece model (Kudo and Richardson, 2018) for the target language vocabulary of size 8,000 (Di Gangi et al., 2020). For the source language vocabulary of size 5,000 we use a BPE SubwordNMT model (Sennrich et al., 2016) with Moses tokenizer (Koehn et al., 2007). The reason for which we used SubwordNMT instead of SentencePiece lies in the strategy used for determining the end of a word, which is crucial for simultaneous inference. While SentencePiece uses the character "_" at the beginning of a new word, SubwordNMT appends "@@" to any token that does not represent the end of a word. Thus, SentencePiece units require the generation of the first token of the next word to determine if the current word is over while SubwordNMT units do not. For instance, the sentence "this is a phrase", is encoded into SentencePiece units as "_th is _is _a _ph rase". As such, to determine if "_th is" is a complete word, we have to wait for the next word with the "_" character at the beginning, that is "_is". Instead, with SubwordNMT we have "th@@ is is a ph@@rase ", and we do not need to receive "is" to determine that "th@@ is" is finished.

We select the best checkpoint based on the loss and early stop the training if the loss did not improve for 10 epochs. We trained the system for 100 epochs at maximum. At the end of the training, we make the average of the 7 checkpoints around the best one.

For the inference part, we use the SimulEval tool (Ma et al., 2020a) as in (Ma et al., 2020b) with the additional `force_finish` tag that forces the model to generate text until the source speech has been completely ingested, i.e. to ignore the end of sentence token if predicted before the end of an utterance. In case of *wait-k* with adaptive word detection, we also force the model to predict the successive most probable token if the end of sentence is predicted (that we called `avoid_eos_while_reading`), while for the fixed we found that it degrades the performance. The detection is taken every average word duration, that is every $280ms$, as estimated by Ma et al. (2020b) in the MuST-C dataset.

A.2 Conformer

For the Conformer model, we build an architecture similar to Inaguma et al. (2021), we use 12 Conformer encoder layers and 6 Transformer decoder layers. The number of parameters is $\sim$ 35.7M. We use the same embedding dimension of our Transformer-based architecture, 4 attention encoder heads and 8 attention decoder heads. For the Conformer Feed-Forward layer, Attention layer, and Convolution layer, we use 0.1 as dropout. We use a kernel size of 31 for the point- and depth-wise convolutions of the Convolution layer. The vocabularies are the same of the Transformer-based, as well as the selection of the checkpoint. At inference time, the `force_finish` tag is used with the `avoid_eos_while_reading` for both the word detection strategies.

A.3 Machine Translation

The MT model used to generate the target for the KD was trained on OPUS datasets (Tiedemann, 2016). It is a plain Transformer with 16 attention heads and 1024 features in encoder/decoder embeddings, resulting in 212M parameters. The English→German MT scores 32.1 BLEU and the English→Spanish MT scores 35.8 BLEU on MuST-C tst-COMMON.

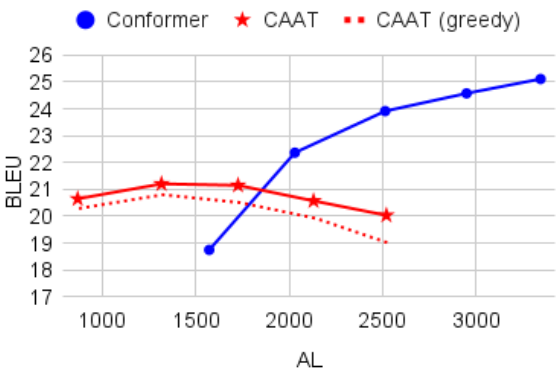
B Under-generation Statistics

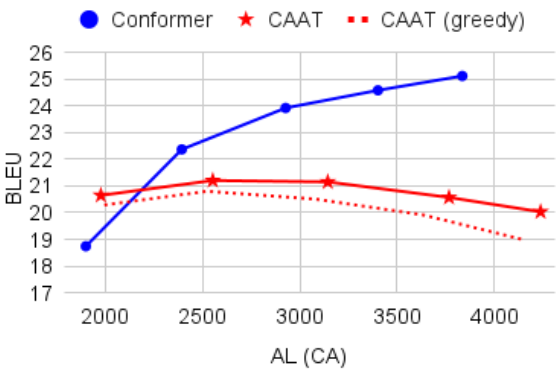
In Section7, while discussing the en-de curves of Figure 5, we highlighted a performance degradation of CAAT at higher latency regimes. In fact, during our experiments we observed that CAAT tends to generate shorter sentences as the value of k increases. This behaviour becomes apparent in Table 2, where we report the word length difference between the generated hypotheses and the corresponding references. For en-de, CAAT exhibits a strong tendency to under-generate (indicated by negative values) at high latency and this is presumably the reason why we observed the BLEU drop.

English→German					
Model	k=3	k=5	k=7	k=9	k=11
Conformer	-1	-0.94	-0.93	-0.77	-0.63
CAAT	0.47	-0.3	-0.79	-1.26	-1.55
English→Spanish					
Model	k=3	k=5	k=7	k=9	k=11
Conformer	0.48	0.49	0.53	0.74	0.80
CAAT	1.57	0.96	0.61	0.35	0.18

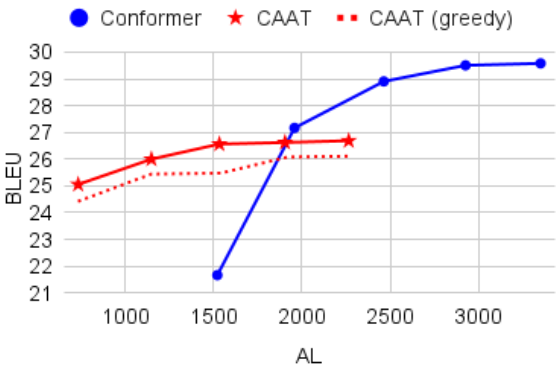
Table 2: Average word length difference w.r.t. the reference. Positive values indicate exceeding words, negative values indicate missing words.

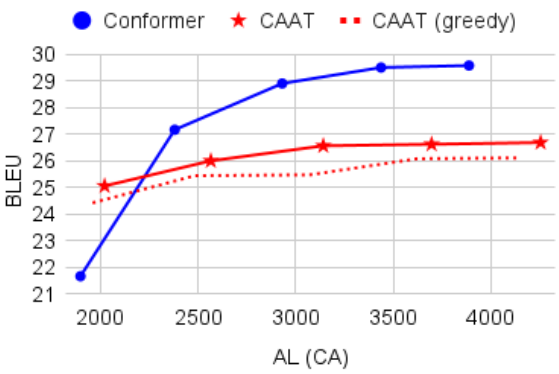
C Average Lagging





(a) English → German





(b) English → Spanish

Figure 6: AL/AL_{CA}-BLEU curves of our offline-trained Conformer and CAAT models.