wav2vec-S: Adapting Pre-trained Speech Models for Streaming

Biao Fu[1,3] [*], **Kai Fan**[2] [†], **Minpeng Liao**[2],
Yidong Chen[1,3], **Xiaodong Shi**[1,3] [†], **Zhongqiang Huang**[2]
[1]School of Informatics, Xiamen University
[2]Alibaba Group
[3]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan (Xiamen University), Ministry of Culture and Tourism
biaofu@stu.xmu.edu.cn,{ydchen,mandel}@xmu.edu.cn
{k.fan,minpeng.lmp,z.huang}@alibaba-inc.com

Abstract

Pre-trained speech models, such as wav2vec 2.0, have significantly advanced speech-related tasks, including speech recognition and translation. However, their applicability in streaming scenarios is limited because these models are trained on complete utterances, leading to a mismatch with incremental streaming inputs. This paper identifies three critical design aspects within the architecture of wav2vec 2.0 and proposes a novel model, wav2vec-S, which incorporates simple modifications to ensure consistent speech representations during both training and inference phases for streaming speech inputs. Furthermore, we demonstrate that wav2vec-S models can be efficiently adapted from pre-trained wav2vec 2.0 models through continued pre-training and effectively finetuned to meet various latency requirements in downstream applications. Experiments on speech recognition and translation tasks show that wav2vec-S outperforms strong baseline models and achieves a superior balance between quality and latency.

1 Introduction

The advent of pre-trained speech models, such as wav2vec (Schneider et al., 2019; Baevski et al., 2020b), HuBERT (Hsu et al., 2021), WavLM (Chen et al., 2022), and data2vec (Baevski et al., 2022, 2023), have catalyzed significant progress in speech-related tasks, including automatic speech recognition (ASR) (Baevski et al., 2020b; Hsu et al., 2021) and speech translation (ST) (Xu et al., 2021; Fang et al., 2022; Wang et al., 2022; Fang and Feng, 2023). These models learn powerful speech representations from large amounts of unlabeled speech data through self-supervised learning techniques, leading to significant performance improvements.

[*] Work was done during Biao Fu's internship at Alibaba.
[†] Corresponding author.

Despite the advancements, their potential has not yet been fully harnessed for streaming scenarios. These models, trained and optimized for processing complete utterances, yield sub-optimal performance when applied to partial speech inputs in streaming scenarios. Previous efforts to bridge this gap have often utilized knowledge distillation techniques (Cao et al., 2021; Shim et al., 2023; Fu et al., 2023a), with the goal of transferring knowledge from offline pre-trained models to streaming student models. However, these strategies require task-specific distillation, and generally involve complex, multi-stage training processes (Cao et al., 2021; Yang et al., 2022) or the need to balance multiple distillation objectives (Shim et al., 2023; Fu et al., 2023b,a). This limits their applicability as universal pre-trained models. Moreover, these methods do not fully exploit the potential of large-scale unlabeled data for streaming speech tasks.

In this paper, we focus on wav2vec 2.0 (Baevski et al., 2020b) and identify three critical design aspects of its architecture that hinder its applicability to streaming tasks. Accordingly, we propose a novel model, wav2vec-S, which incorporates simple modifications to accommodate streaming requirements. Specifically, we substitute layer normalization for group normalization in the feature extractor, shift from convolution-based relative position encoding to absolute position encoding, and implement a block-wise unidirectional self-attention mechanism, all designed to eliminate or restrict dependency on future context, as required for maintaining consistent representations during training and inference for streaming inputs. We demonstrate that instead of training from scratch, wav2vec-S can be efficiently adapted from pre-trained wav2vec 2.0 models through continued pre-training on unlabeled speech using the same self-supervised training objectives. In addition, we introduce mechanisms in both the pre-training stage and the fine-tuning stage to allow the model to meet various latency requirements in downstream tasks.

In summary, our contributions are threefold:

1. We present wav2vec-S, a model adapted from wav2vec 2.0 with straightforward architectural modifications tailored for streaming tasks, which can be efficiently adapted from pre-trained wav2vec 2.0 models through continued pre-training.

2. We conduct comprehensive experiments on two popular streaming speech tasks: ASR and ST, to demonstrate that fine-tuning from wav2vec-S outperforms multiple strong baseline approaches in balancing between quality and latency.

3. We release the source code and trained models to facilitate future research of pretrained models for streaming tasks.[1]

2 Preliminary

Wav2vec 2.0 represents a significant advancement in speech processing and has been widely utilized as a backbone for various speech-related tasks (Yi et al., 2021; Pepino et al., 2021; Fan et al., 2021; Xu et al., 2021; Siuzdak et al., 2022; Popuri et al., 2022). Our proposed model is based on wav2vec 2.0 with essential modifications to support streaming applications. We first briefly review its model architecture and training objectives.

2.1 Model Architecture

The architecture of wav2vec 2.0 includes three main components: a feature encoder, a context network, and a quantization module.

Feature Encoder. The feature encoder converts the raw audio waveform into feature representations. It consists of seven temporal convolutional layers, each followed by group normalization (Wu and He, 2018) and a GELU activation function (Hendrycks and Gimpel, 2023).

Context Network. The context network adopts the Transformer architecture (Vaswani et al., 2017) along with a convolution-based relative position embedding layer. It processes the output of the feature encoder to extract contextualized representations of the speech.

[1]The source code and trained models are available at https://github.com/biaofuxmu/wav2vec-S

Quantization Module. This module employs vector quantization to map the continuous feature representations from the feature encoder to discrete tokens via a codebook and a Gumbel-Softmax operation (Jang et al., 2016). These discrete tokens are used as target labels for self-supervised learning of the entire network.

2.2 Training Objectives

The training of wav2vec 2.0 integrates both contrastive loss and diversity loss, as follows:

$$\mathcal{L} = \mathcal{L}_c + \alpha \mathcal{L}_d, \tag{1}$$

Contrastive Loss. The loss $\mathcal{L}_c$ leverages the contextualized representation c_t from the context network, based on feature representations masked at time step t. It aims to differentiate the true quantized vector q_t from a set of negative samples $\bar{\mathbf{q}}_t = \bar{q}_1, \bar{q}_2, \ldots, \bar{q}_k$, sampled from different masked time steps of the same utterance. The loss is calculated as follows:

$$\mathcal{L}_c = -\sum_t \log \frac{\exp\left(\mathrm{sim}\left(c_t, q_t\right)/\tau\right)}{\sum_{\bar{q} \in \bar{\mathbf{q}}_t} \exp\left(\mathrm{sim}\left(c_t, \bar{q}\right)/\tau\right)} \tag{2}$$

Diversity Loss. The loss $\mathcal{L}_d$ encourages the model to utilize the entire spectrum of quantized speech representations, by maximizing the entropy of the averaged distribution of codebook entries.

3 Our Proposal: wav2vec-S

We next describe how to adapt wav2vec 2.0 to wav2vec-S for streaming, focusing on three key aspects: architecture, pre-training, and fine-tuning.

3.1 Architectural Modifications

We identify three critical design aspects within the architecture of wav2vec 2.0 that hinder its application to streaming scenarios. We propose simple modifications to preserve its modeling capabilities while enabling streaming processing.

3.1.1 Feature Normalization

In the wav2vec 2.0 architecture, group normalization (GN) (Wu and He, 2018) is applied within the feature encoder, where the normalization process calculates the mean and standard deviation across both the sequence and feature dimensions of complete utterances. However, this methodology results in sub-optimal performance for streaming applications, in which only partial utterances are available during real-time processing, due to the difference in statistical computation between the training and inference stages.

To address this issue, we utilize layer normalization (LN) (Ba et al., 2016), which determines the mean and standard deviation exclusively along the feature dimension, disregarding the sequence length. This ensures consistent processing, whether dealing with complete or partial utterances. While other normalization techniques like weight normalization (Salimans and Kingma, 2016), employed in Encodec (Défossez et al., 2022) for streaming models, may also be viable, since it reparametrizes the layer's parameters instead of altering the data values. We leave the investigation of various normalization methods for future research.

3.1.2 Positional Encodings

wav2vec 2.0 incorporates implicit relative positional encodings (PE), as detailed in (Mohamed et al., 2020), within its context network. These encodings are generated through a 1D convolution layer with a substantial kernel size, representing 2560ms of speech. This extensive context size necessitates significant latency to accurately compute positions at the most recent segments of streaming input, without resorting to padding, to maintain alignment with the training on complete utterances.

To overcome these challenges, wav2vec-S adopts absolute sinusoidal PE (Vaswani et al., 2017) for its context network.[2] This method ensures stable positional information, regardless of the availability of future context, making it suitable for streaming applications.

3.1.3 Self-Attention

Despite its strong modeling capabilities by accessing both past and future contexts, the use of bidirectional self-attention in wav2vec 2.0 presents two limitations for streaming applications. Firstly, the absence of future context for the latest segment in streaming input creates a mismatch between training on complete utterances and inference on partial ones. Secondly, past segments require repeated computation of their bidirectional self-attentions as their future context expands. Although unidirectional self-attention, as investigated in prior studies (Ren et al., 2020; Ma et al., 2020b), avoids these pitfalls by only leveraging past context, it results in degraded performance due to the lack of utilizing any future context. Research has shown that incorporating even a small amount of future context can markedly enhance the quality of streaming speech features with minimal latency (Fu et al., 2023a).

In response, we employ block-wise self-attention (Liu et al., 2021), which allows for the utilization of all past context and a limited amount of future context. Specifically, the hidden states in the l-th layer, denoted as $\mathbf{h}^l_{0:T-1}$, are segmented into m-sized non-overlapping blocks, $\mathbf{b}^l_i = \mathbf{h}^l_{m\times i:m\times(i+1)-1}$, and are calculated in a block-wise manner. The input hidden states at the $(l-1)$-th layer are partitioned into overlapping blocks, adopting a block processing strategy (Tsunoo et al., 2019; Liu et al., 2021) with a stride of m and a block size of $m+r$, under the constraint that $r \leq m/2$. This results in two parts for each block: the main block $\mathbf{b}^{l-1}_i = \mathbf{h}^{l-1}_{m\times i:m\times(i+1)-1}$ and its right-side future context $\mathbf{r}^{l-1}_i = \mathbf{h}^{l-1}_{m\times(i+1):m\times(i+1)+r-1}$, which overlaps with the next block.

To compute the hidden states in each block $\mathbf{b}^l_i$ of the l-th layer, the query, key, and value matrices are calculated by facilitating bidirectional attention within blocks and unidirectional attention across blocks, as follows:

$$\begin{aligned}
\mathbf{q}^{l-1}_i &= \mathbf{W}_q \left[\mathbf{b}^{l-1}_i, \mathbf{r}^{l-1}_i\right] \\
\mathbf{k}^{l-1}_i &= \mathbf{W}_k \left[\mathbf{b}^{l-1}_0, \cdots, \mathbf{b}^{l-1}_i, \mathbf{r}^{l-1}_i\right] \\
\mathbf{v}^{l-1}_i &= \mathbf{W}_v \left[\mathbf{b}^{l-1}_0, \cdots, \mathbf{b}^{l-1}_i, \mathbf{r}^{l-1}_i\right]
\end{aligned}$$

3.2 Pre-training

The proposed wav2vec-S model can be pretrained from scratch using a substantial volume of unlabeled speech data, similarly to the wav2vec 2.0 model. However, such a process demands considerable computational resources and time, comparable to the efforts required for training other large pretrained speech models. Given its significant architectural resemblance to wav2vec 2.0, a more efficient strategy is to begin with the pretrained weights from wav2vec 2.0 and continue pre-training on unlabeled speech data, adhering to the same training loss formula presented in Eq. (1). This method leverages the extensive computational work previously applied to the wav2vec 2.0 model and its learned complex speech patterns, enabling the wav2vec-S model to be trained more efficiently with significantly fewer updates. This approach

[2]There are several alternative solutions for PE, such as relative PE. Considering our main goal is to adapt the model for streaming use cases rather than to assess the effectiveness of different PE techniques, we opted for absolute PE for its simplicity, though the other approaches were also viable. We leave the study of various PE methods for future research.

is validated by the empirical results discussed in Section 4.6.

The size of blocks in block-wise self-attention directly impacts the latency of the pre-trained model. Utilizing a fixed block size during pre-training could limit performance in streaming applications with diverse latency requirements, as will be demonstrated in Section 4.5. To make the pre-trained wav2vec-S model versatile in streaming scenarios with varied latency needs, we implement a mechanism to dynamically adjust the block size during the pre-training stage. Specifically, the sizes m of the main context and r of the future context are randomly selected during each training step from the ranges of $[160\text{ms}, 640\text{ms}]$ and $[80\text{ms}, 320\text{ms}]$, respectively, at increments of 40ms, with the constraint that $r \leq \frac{m}{2}$.

3.3 Fine-tuning

In this work, we focus on streaming speech-to-text tasks as downstream applications, pairing wav2vec-S for parameter initialization of the speech encoder with a separate text decoder. Following recent research trends in this field (Liu et al., 2021; Tang et al., 2023), we primarily apply the Cross Attention Augmented Transducer (CAAT) (Liu et al., 2021) as the decoder due to its state-of-the-art performance. CAAT extends the Recurrent Neural Network Transducer (RNN-T) (Graves, 2012) to support streaming tasks involving reordering, such as speech-to-text translation. For details of the CAAT architecture, please see (Liu et al., 2021).

Central to the design of CAAT is its goal to jointly optimize a READ/WRITE policy and a translation model. Latency management within the CAAT framework is facilitated through a hyper-parameter known as the decision step d, which dictates that read-write decisions are made every d blocks. In the original CAAT framework, to achieve optimal performance, the d value selected during inference must match the value used during training, leading to the increased cost of training and maintaining multiple models to accommodate different latency requirements. To mitigate this challenge, we propose randomly selecting different d values for each mini-batch within a single training session, akin to the dynamic block size approach in pre-training discussed in Section 3.2. This strategy enables training a single fine-tuned model capable of meeting a wide range of latency demands.

In addition we also adapt wav2vec-S to other streaming models that use wav2vec 2.0 as a speech encoder to verify its generalization, as will be demonstrated in Section 4.4.

4 Experiments

4.1 Experiments Settings

Datasets For continued pre-training, we use the 960 hours of unlabeled speech data from LibriSpeech[3] (Panayotov et al., 2015) or the 60k hours of unlabeled speech data from LibriVox (Kahn et al., 2020). For streaming ASR fine-tuning, we use the 960 hours of labeled speech data from LibriSpeech dataset as the training set. For streaming ST fine-tuning, we use the MuST-C v1[4] English-German (EnDe) and English-Spanish (EnEs) datasets (Di Gangi et al., 2019). Detailed data statistics are given in Appendix A. Our models are tuned on the development sets of LibriSpeech or MuST-C.

Pre-Training Settings We pre-trained two variants for wav2vec-S: BASE and LARGE, both of which adhere precisely to the model configurations of their wav2vec 2.0 counterparts. In terms of training settings, our models align with the wav2vec 2.0 (Baevski et al., 2020b) settings, with the sole modification being the reduction of the warmup steps to 5000 due to its initialization with wav2vec 2.0 pre-trained weights. Our implementation is based on Fairseq[5] (Ott et al., 2019).

Finetuning Settings The streaming ASR and ST models share the same model configuration. As with our pre-trained model, we test two model configurations: BASE and LARGE. The BASE and LARGE models contain 6 and 12 Transfomer decoder layer, respectively, and maintain consistency in the decoder hidden size with their respective pre-trained encoders. The total number of parameters for our pretrained and fine-tuned models see Appendix B. More training details see Appendix C.

Baseline Models We compare our method with several strong baselines that have a similar number of parameters.

CAAT (Liu et al., 2021) utilizes an RNN-T style architecture with transformer layers. CAAT BASE and LARGE models have similar hidden dimensions and layers to our model configurations, with

[3] https://www.openslr.org/12
[4] https://ict.fbk.eu/must-c
[5] https://github.com/pytorch/fairseq

the exception that they do not employ pre-trained models learned from unlabeled data.

MU-ST (Zhang et al., 2022) trains an offline ST model and then optimizes a speech segmenter to learn an adaptive segmentation policy to detect meaningful units. MU-ST BASE is trained from scratch, and MU-ST LARGE uses the pre-trained wav2vec LARGE and mBART50 to initialize their model and then fine-tunes with external ST data.

FAST (Fu et al., 2023a) uses wav2vec 2.0 as the speech encoder to train an offline ST model and then introduces future-aware distillation and future-aware inference to mitigate the mismatch between offline training and streaming inference, adapting this offline model to streaming ST.

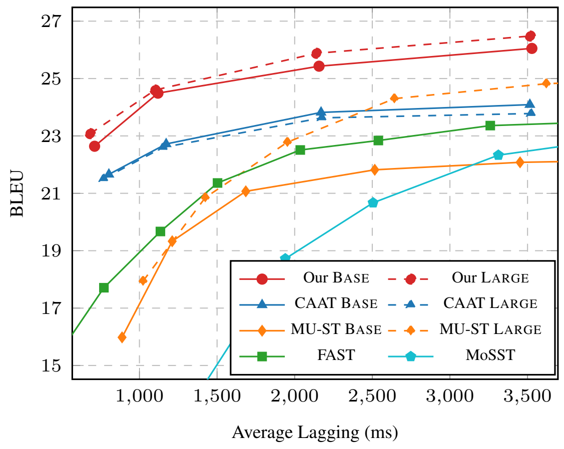
MoSST (Dong et al., 2022) applies wav2vec 2.0 as the speech encoder with a monotonic segmentation module to train an offline ST model for streaming inference.

Evaluation We average the checkpoints of the best 10 epochs on development set for evaluation (Tang et al., 2023; Fu et al., 2023a). We measure the ASR quality by the word error rate (WER) and apply a text normalizer[6] (Radford et al., 2023) to standardize text before the WER calculation. We use SacreBLEU[7] to compute case-sensitive detokenized BLEU for the translation quality. The latency[8] is evaluated with Average Latency (AL) (Ma et al., 2019) in the SimulEval[9] (Ma et al., 2020a).

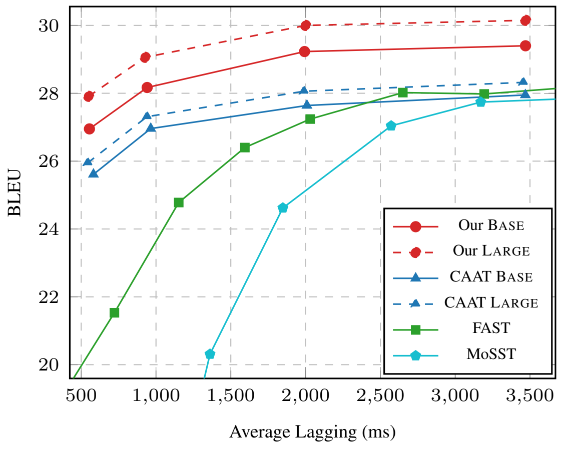
4.2 Streaming Results

Streaming ST We presents the streaming ST results in Figure 1. Compared to the CAAT BASE and MU-ST BASE models without self-supervised pre-training, our BASE model achieves higher BLEU in all latency regions. There is an improvement of over 1 BLEU score in both EnDe and EnEs. Other methods, such as MoSST, using original wav2vec 2.0 as the speech encoder, show inferior translation quality in low and medium latency areas due to the mismatch between bidirectional pre-training and streaming inference. While FAST alleviates this mismatch to some extent, it still exhibits a significant performance gap compared to our method.

[6]`https://github.com/openai/whisper`
[7]`https://github.com/mjpost/sacrebleu`
[8]The extended results for other latency metrics (Average Proportion (AP) (Cho and Esipova, 2016) and Differentiable Average Lagging (DAL) (Cherry and Foster, 2019)) are described in Appendix D.3.
[9]`https://github.com/facebookresearch/SimulEval`



(a) EnDe

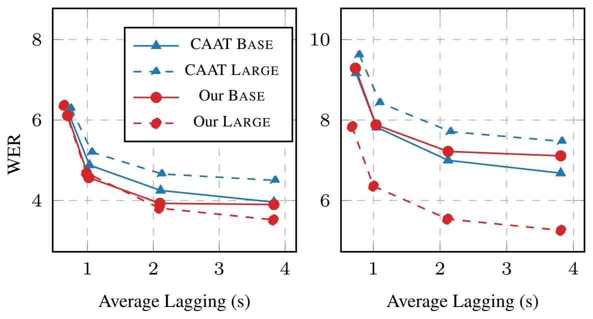


(b) EnEs

Figure 1: The translation quality (BLEU) against the latency metrics (AL) on the `tst-COMMON` set of MuST-C EnDe and EnEs datasets.

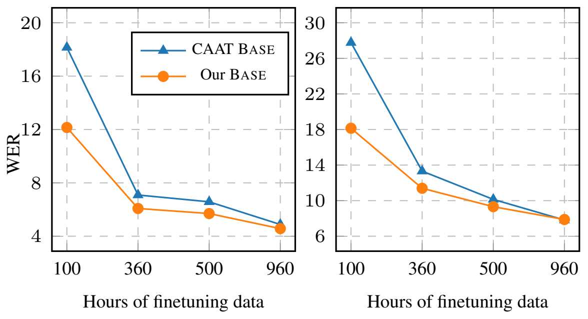
When scaling the models to the LARGE size, CAAT LARGE shows no improvement in translation quality for EnDe, whereas for EnEs, the improvement is marginal. This can be attributed to overfitting due to the absence of pre-training. MU-ST LARGE, which directly uses bidirectional wav2vec 2.0, only improves in high latency scenarios, with no gains in low latency. In contrast, our LARGE model shows significant improvements in all latency regions, with approximately 0.4 BLEU in EnDe and about 0.8 BLEU in EnEs.

Streaming ASR Figure 2 illustrates the results of streaming ASR on the LibriSpeech `test-clean` and `test-other` sets when finetuning on the LibriSpeech 960h labeled data. Our BASE slightly outperforms the CAAT BASE across all latency regions on `test-clean` set and achieves comparable performance on `test-other` set. The relatively small difference between two BASE mod-



(a) test-clean (b) test-other

Figure 2: ASR quality (WER) against the latency metrics (AL) on the LibriSpeech test sets when fine-tuning on the LibriSpeech 960h data.



(a) test-clean (b) test-other

Figure 3: ASR quality (WER) against the finetuing data size on the LibriSpeech test sets.

els may be attributed to two primary factors: firstly, the BASE model was pre-trained solely on the LibriSpeech 960h data, without introducing additional data compared to the fine-tuned data; secondly, the large quantities of fine-tuning data leveraged may mitigate performance gains.

Upon scaling to the LARGE configuration, which benefits from pre-training on an large-scale Libri-Light 60k hours of unlabeled data, we observe more substantial performance gains. Our LARGE model not only achieves lower WER compared to its BASE counterpart but also maintains a significant advantage over the CAAT LARGE model. However, we observe that the performance of the CAAT LARGE model becomes inferior when scaled up. As in Section 4.2, this is attributed to overfitting caused by the lack of pre-training on large-scale unlabeled data.

To further explore the second hypothesis regarding the influence of fine-tuning data size, Figure 3[10] compares the streaming ASR performance of

Model	EnDe	EnEs
Fairseq ST (Wang et al., 2020a)	22.70	27.20
ESPnet-ST (Inaguma et al., 2020)	22.91	27.96
Wav2vec2-Transformer (Ye et al., 2021)	24.15	28.10
wav2vec-S-Transformer (Ours)	23.82	28.24

Table 1: The performance of offline ST on the MuST-C tst-COMMON set.

Model	test-clean	test-other
ContextNet (Han et al., 2020)	2.1	4.6
Conformer (Gulati et al., 2020)	2.1	4.3
wav2vec 2.0 BASE (Baevski et al., 2020b)	3.4	8.5
wav2vec 2.0 LARGE (Baevski et al., 2020b)	2.2	4.5
wav2vec-S BASE	4.5	12.6
wav2vec-S LARGE	2.9	6.6

Table 2: The performance of offline ASR finetuned on LibriSpeech 960h labeled data and evaluated the LibriSpeech test sets without LM or lexicon. The results of baselines are obtained from the corresponding papers.

our BASE and CAAT BASE models[11] when fine-tuning on the 960h, 500h, 360h and 100h subsets of Librispeech. As the fine-tuning data is scaled down from the full 960h to a more constrained 100h, we witness a striking enhancement in the performance gains of our BASE model relative to the CAAT BASE. Notably, when fine-tuned with just 100h of data, our BASE model achieves a remarkable increase in WER by ~6 absolute points on the `test-clean` subset and ~10 absolute points on the `test-other` subset. These results suggest the effectiveness of our model in scenarios with limited fine-tuning data due to pre-training on large-scale unlabeled data.

4.3 Offline Results

Besides the streaming speech-to-text tasks, we also verify the capability of wav2vec-S on offline tasks.

Offline ST We implement a baseline ST model Wav2vec2-Transformer following Ye et al. (2021), which integrates wav2vec 2.0 with a two-layer CNN and a Transformer. Our offline ST model follows the same configuration as the baseline, except the original wav2vec 2.0 component is replaced by our wav2vec-S to investigate its performance in a non-streaming setting, where the block size is set to 960ms (m = 640ms and r = 320ms). Table 1 reports the comparison results on MuST-

[10]For a fair comparison, we try our best to set the AL of different data size to be approximately equal (AL$\approx$ 1000ms). The complete latency-quality curves for different data sizes are shown in the Appendix D.2.

[11]Since our LARGE model has achieved significant gains, we only compare the performance of the BASE models in this experiment.

C EnDe and EnEs datasets. In comparison to Fairseq-ST and ESPnet-ST, which do not apply pre-training, wav2vec based models show significant better performance. When compared to the Wav2vec2-Transformer, our model shows a decrease in EnDe by a narrow margin of 0.33 BLEU but an increase for EnEs by 0.14 BLEU. We hypothesize that translation performance disparities among language pairs are due to syntactic variances in word order. Our wav2vec-S's block-size self-attention, limited by a narrow contextual window, struggles with long-range dependencies. This limitation hampers translations into languages like German with subject-object-verb (SOV) order, unlike English's subject-verb-object (SVO) order. However, for languages with similar SVO word order, like Spanish, wav2vec-S's localized context suffices for precise translation. These results demonstrate that our wav2vec-S, even when applied in an offline setting, maintains competitive performance against established offline models.

Offline ASR We strictly follow wav2vec 2.0 (Baevski et al., 2020b) to fine-tune our offline ASR models on LibriSpeech 960h data with connectionist temporal classification (CTC) loss (Graves and Graves, 2012). The block size is consistent with the settings used in our offline ST model. Table 2 shows the performance of offline ASR models without a language model (LM) or lexicon in decoding. wav2vec-S BASE model exhibits an increase in Word Error Rate (WER) by 1.1 on the `test-clean` set and by 4.1 on the `test-other` set compared to wav2vec 2.0, This reduction in performance is largely due to the block-size self-attention mechanism used in wav2vec-S, which is tailored for streaming scenarios but struggles with modeling the long-term dependencies. When scaling up to the LARGE configuration and pretraining on more unlabel data, the performance gap narrows. Despite the decrease in performance when compared to wav2vec 2.0, it is imperative to recognize that wav2vec 2.0's architecture is not inherently suited for streaming applications.

4.4 Adapting to Other Streaming ST Models

To validate the generalizability of wav2vec-S, we extended its application to other streaming ST methods, including MoSST and ITST, which traditionally employ wav2vec 2.0 directly. Figure 4 presents a comprehensive comparison of ITST, MoSST, and our adapted versions of these models, which we refer to as wav2vec-S-ITST and wav2vec-S-MoSST, respectively. We employ AL represented by solid curves, and AL_CA depicted by dashed curves to measure ideal and computation-aware latency, respectively.

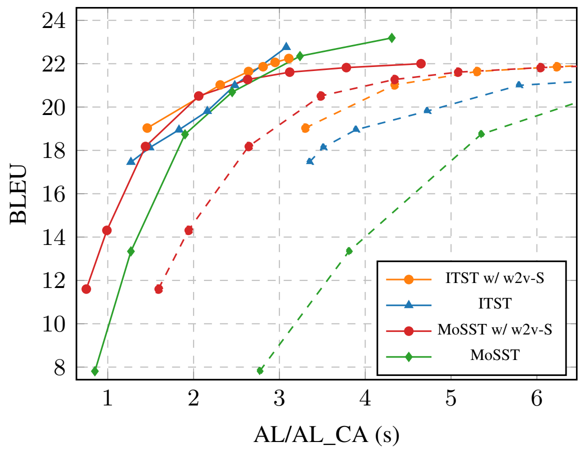


Figure 4: Comparison with ITST and MoSST on MuST-C EnDe `tst-COMMON` set. Solid curves represent AL, dashed curves represent AL_CA.

Our adapted models, wav2vec-S-MoSST and wav2vec-S-ITST, outperform the original MoSST and ITST models in both low and high latency settings (AL < 2s), with an improvement of approximately 4 BLEU for MoSST and 1 BLEU for ITST. These results indicate that our adjustments to wav2vec 2.0 benefit streaming speech-to-text (ST) tasks. Considering computational latency (AL_CA), our models demonstrate a significant reduction in latency due to eliminating the need for re-encoding speech frames, unlike the original models. Overall, our experiments confirm that wav2vec-S delivers both high translation quality and reduced computational latency.

4.5 Ablation Study

To study the effectiveness of our methods, we compare our BASE model with several variants in the streaming ST task[12]. The results are shown in Figure 5. All ablation results are evaluated on the MuST-C EnDe `tst-COMMON` set.

To validate the effectiveness of our proposed modifications, we sequentially remove three key modifications. The experimental results, as illustrated in Figure 5, include the following variants: "w/o FN modification" (**removal of the feature normalization modification**), "w/o FN and PE

[12]Unless otherwise specified, the BASE configuration is used in the subsequent experiments.

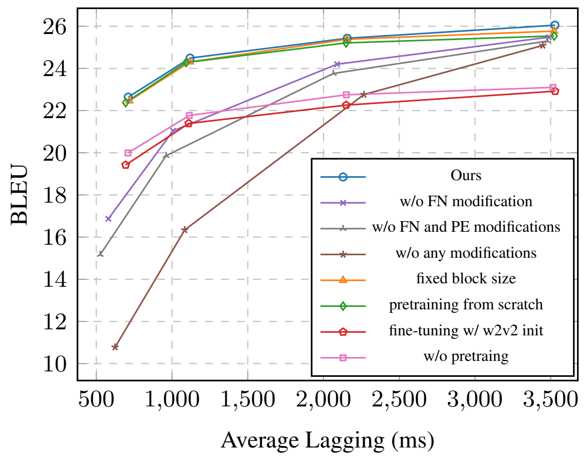


Figure 5: Ablation study of our approach on the `tst-COMMON` set of MuST-C EnDe dataset. FN: feature normalization, PE: position encoding.

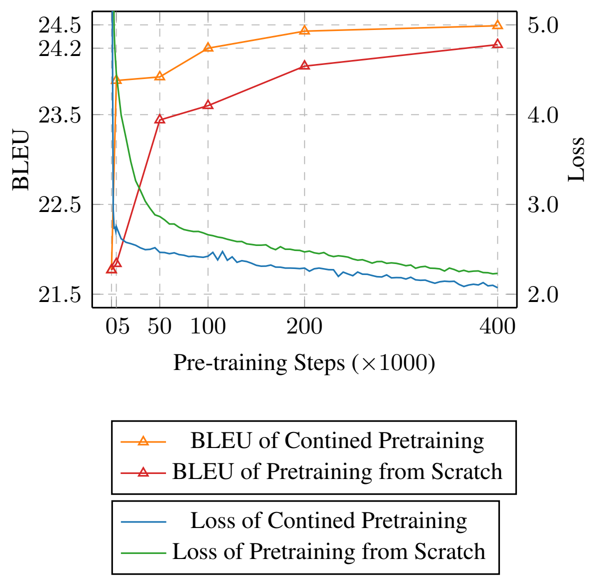


Figure 6: BLEU scores and loss values over pre-training steps. We evaluate the BLEU scores on the MuST-C EnDe `tst-COMMON` set and present the loss values on the validation set during pre-training. For fair comparison, the six points in the BLEU curve have similar AL values, all around 1100ms.

modifications" (**removal of both the feature normalization and positional encoding modifications**), and "w/o any modifications" (**removal of all three modifications**). We observe that each variant performs significantly worse than "Ours" model in the low latency region. Moreover, the streaming translation performance progressively deteriorates as the modifications are incrementally removed. These results indicate that all three modifications must be implemented to adequately support streaming applications; otherwise, a training/inference mismatch problem may occur.

Subsequently, we investigate appropriate streaming settings to explore the effects of different model configurations. Specifically, we evaluate:
(1) **The impact of dynamic versus fixed block size.** Our results, as shown in Figure 5, demonstrate that a dynamic block size ("Ours") can adapt to different latency settings without compromising performance, achieving consistent improvement over the "fixed block size" model[13].
(2) **The effects of different training strategies.** We compare the approach of pre-training wav2vec-S from scratch versus continued pre-training with parameters initialized from wav2vec 2.0. The superior performance of "Ours" compared to the "pre-training from scratch" model, as illustrated in Figure 5, underscores the benefits of the continued pre-training strategy.
(3) **The impact of using different pre-trained models for downstream streaming ST task.** In Figure 5, we contrast the performance of a streaming model (with the same encoder architecture as wav2vec-S) fine-tuned from wav2vec-S ("Ours") against models fine-tuned from wav2vec 2.0 ("fine-tuning w/ w2v2 init") or without a pre-trained model ("w/o pretraining"). The results show that "Ours" consistently outperforms others across all latency regions, demonstrating the advantage of pre-trained streaming models.

In summary, these results demonstrate the contribution of our pretraining method and the superiority of streaming pretraining for streaming tasks.

4.6 Analysis on Continued Pre-training

In this section, we investigate the pretraining efficiency of our continued pre-training strategy. Figure 6 illustrates the changes of BLEU scores and loss values with pre-training steps for continued pre-training and pre-training from scratch. Firstly, the continued pre-training approach exhibits a notably faster convergence rate in terms of loss reduction, signifying a more efficient optimization process. Moreover, the loss values associated with continued pre-training remain lower throughout the training process compared to pretraining from scratch. This suggests that continuous pre-training not only accelerates the convergence process, but also enhances the model's overall optimization, leading to better performance. Additionally, the

[13]The block size is fixed to 480 ms during pre-training ($m = 320$ ms, $r = 160$ ms).

continued pre-training strategy achieves a comparable BLEU score to the pre-training from scratch with significantly fewer training steps. For instance, at the 100k training steps, the continued pre-training model exhibits a BLEU score improvement that the pre-training from scratch only matches at a later stage, up to 400k steps. These results demonstrate the efficiency of our continued pretraining.

5 Related Works

Pre-trained Speech models Existing work can be broadly categorized by the pre-training objective into three main classes: contrastive learning-based models (Schneider et al., 2019; Baevski et al., 2020a,b), mask prediction-based models (Hsu et al., 2021; Chen et al., 2022; Baevski et al., 2022), and masked autoencoder-based models (Gong et al., 2022; Huang et al., 2022; Baevski et al., 2023; Chen et al., 2023). However, these methods cannot be effectively applied to streaming scenarios as they are essentially designed for offline tasks.

Streaming ST This method consists of a speech translation model and a read-write policy. Existing research primarily focuses on the exploration of the read-write policy, categorizing them into fixed (Ren et al., 2020; Ma et al., 2020b; Zeng et al., 2021; Papi et al., 2022; Wang et al., 2023) and adaptive policy (Liu et al., 2021; Zhang and Feng, 2022, 2023b; Papi et al., 2023a,b; Zhang et al., 2023a; Zhao et al., 2023; Chen et al., 2024). Additionally, recent studies leverage the wav2vec 2.0 model to enhance the quality of the ST model (Dong et al., 2022; Zhang and Feng, 2022, 2023a; Zhang et al., 2023b; Yang et al., 2023). While significant performance improvements are achieved at higher latency region, this integration has resulted in poorer performance at lower and medium latency regions due to the mismatch between offline training and streaming inference (Fu et al., 2023a). To this end, some research efforts introduces various distillation losses (Fu et al., 2023a) or multi-stage training objectives (Zhang et al., 2023a), which have been effective in mitigating the mismatch to some extent.

Streaming ASR Current methods can be broadly classified into transducer-based approaches, such as RNN-T (Graves, 2012), Neural Transducer (Jaitly et al., 2015), and Transformer Transducer (Yeh et al., 2019), and local attention-based methods, including monotonic attention (Raffel et al., 2017; Ma et al., 2020c), chunk/block-wise attention (Chiu and Raffel, 2018; Zhang et al., 2020; Wang et al., 2020b). Additionally, some studies apply wav2vec 2.0 or large amounts of unlabeled data to streaming ASR (Cao et al., 2021; Yang et al., 2022; Shim et al., 2023; Fu et al., 2023b). However, such approaches often involve complex training procedures and require intricate balancing of multiple objectives, limiting their application to other streaming tasks.

Distinctively, wav2vec-S, through streaming pre-training, addresses above issues, achieving significant improvements across all latency regions.

6 Conclusion

In this paper, we propose a streaming pre-trained speech model, wav2vec-S, which is effectively adapted from the pretrained wav2vec 2.0 model for streaming tasks through simple architectural modifications and continued pretraining. Experiments on both streaming ASR and ST tasks show the superiority of wav2vec-S in terms of performance, pre-training and inference efficiency.

Acknowledgements

We would like to thank all the anonymous reviewers for the insightful and helpful comments. This work is supported by National Science and Technology Major Project (Grant No. 2022ZD0116101), the Key Support Project of NSFC-Liaoning Joint Foundation (Grant No. U1908216), and the Project of Research and Development for Neural Machine Translation Models between Cantonese and Mandarin (Grant No. WT135-76) and the Major Scientific Research Project of the State Language Commission in the 13th Five-Year Plan (Grant No. WT135-38), University-Industry Cooperation Programs of Fujian Province of China (No. 2023H6001), and Alibaba Research Intern Program.

Limitation

Our method primarily focus on adapting the wav2vec 2.0 framework for streaming speech inputs. While wav2vec 2.0 serves as a robust foundation due to its proven effectiveness in various speech-related tasks, our approach does not extend to other pre-trained speech models such as HuBERT (Hsu et al., 2021). However, these pretrained models share an identical or similar Transformer architecture as wav2vec 2.0. Thus, the principles behind the modification strategies we applied to wav2vec 2.0 can also be extended to these models, with necessary changes in some cases. We focus on wav2vec 2.0 serves as a case study to illustrate the applicability and effectiveness of adapting pretrained models for streaming applications. Additionally, the lack of computing resources also prevented us from conducting comprehensive studies using other pretrained models. We found that focusing on wav2vec 2.0 strikes a good balance as it allows us to directly compare with baseline methods that are based on wav2vec 2.0. We leave the extension to other pre-trained models for future research.

Our evaluation of the proposed wav2vec-S model is confined to ASR and ST tasks. While these tasks are critical benchmarks for spoken language understanding, our method has not been validated across a wider spectrum of speech-related tasks, such as speech-to-speech task.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexei Baevski, Arun Babu, Wei-Ning Hsu, and Michael Auli. 2023. Efficient self-supervised learning with contextualized target representations for vision, speech and language. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 1416–1429. PMLR.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. 2022. data2vec: A general framework for self-supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1298–1312. PMLR.

Alexei Baevski, Steffen Schneider, and Michael Auli. 2020a. vq-wav2vec: Self-supervised learning of discrete speech representations. In *International Conference on Learning Representations*.

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

Songjun Cao, Yueteng Kang, Yanzhe Fu, Xiaoshuo Xu, Sining Sun, Yike Zhang, and Long Ma. 2021. Improving Streaming Transformer Based ASR Under a Framework of Self-Supervised Learning. In *Proc. Interspeech 2021*, pages 706–710.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. 2022. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518.

Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, Wanxiang Che, Xiangzhan Yu, and Furu Wei. 2023. BEATs: Audio pre-training with acoustic tokenizers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5178–5193. PMLR.

Xinjie Chen, Kai Fan, Wei Luo, Linlin Zhang, Libo Zhao, Xinggao Liu, and Zhongqiang Huang. 2024. Divergence-guided simultaneous speech translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17799–17807.

Colin Cherry and George Foster. 2019. Thinking slow about latency evaluation for simultaneous machine translation. *arXiv preprint arXiv:1906.00048*.

Chung-Cheng Chiu and Colin Raffel. 2018. Monotonic chunkwise attention. In *International Conference on Learning Representations*.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. 2022. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*.

Zhiyun Fan, Meng Li, Shiyu Zhou, and Bo Xu. 2021. Exploring wav2vec 2.0 on speaker verification and language identification. *arXiv preprint arXiv:2012.06185*.

Qingkai Fang and Yang Feng. 2023. Understanding and bridging the modality gap for speech translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15864–15881, Toronto, Canada. Association for Computational Linguistics.

Qingkai Fang, Rong Ye, Lei Li, Yang Feng, and Mingxuan Wang. 2022. STEMM: Self-learning with speech-text manifold mixup for speech translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7050–7062, Dublin, Ireland. Association for Computational Linguistics.

Biao Fu, Minpeng Liao, Kai Fan, Zhongqiang Huang, Boxing Chen, Yidong Chen, and Xiaodong Shi. 2023a. Adapting offline speech translation models for streaming with future-aware distillation and inference. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16600–16619, Singapore. Association for Computational Linguistics.

Yanzhe Fu, Yueteng Kang, Songjun Cao, and Long Ma. 2023b. Distillw2v2: A small and streaming wav2vec 2.0 based asr model. *arXiv preprint arXiv:2303.09278*.

Yuan Gong, Cheng-I Lai, Yu-An Chung, and James Glass. 2022. Ssast: Self-supervised audio spectrogram transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10699–10709.

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

Alex Graves and Alex Graves. 2012. Connectionist temporal classification. *Supervised sequence labelling with recurrent neural networks*, pages 61–93.

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.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. 2020. ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context. In *Proc. Interspeech 2020*, pages 3610–3614.

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus).

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Po-Yao Huang, Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, and Christoph Feichtenhofer. 2022. Masked autoencoders that listen. In *Advances in Neural Information Processing Systems*, volume 35, pages 28708–28720. Curran Associates, Inc.

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. Association for Computational Linguistics.

Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2015. A neural transducer. *arXiv preprint arXiv:1511.04868*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen, T. Likhomanenko, G. Synnaeve, A. Joulin, A. Mohamed, and E. Dupoux. 2020. Libri-light: A benchmark for asr with limited or no supervision. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7669–7673.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Dan Liu, Mengge Du, Xiaoxi Li, Ya Li, and Enhong Chen. 2021. 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.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020a. SIMULEVAL: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 144–150, Online. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020b. SimulMT to SimulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint*

Conference on Natural Language Processing, pages 582–587, Suzhou, China. Association for Computational Linguistics.

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

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. 2020. Transformers with convolutional context for asr.

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

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022. Does simultaneous speech translation need simultaneous models? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 141–153, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sara Papi, Matteo Negri, and Marco Turchi. 2023a. Alignatt: Using attention-based audio-translation alignments as a guide for simultaneous speech translation. In *Proc. of Interspeech 2023*, Dublin, Ireland.

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

Leonardo Pepino, Pablo Riera, and Luciana Ferrer. 2021. Emotion recognition from speech using wav2vec 2.0 embeddings. In *Proc. Interspeech 2021*, pages 3400–3404.

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

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2837–2846. PMLR.

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.

Tim Salimans and Durk P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.

Kyuhong Shim, Jinkyu Lee, Simyoung Chang, and Kyuwoong Hwang. 2023. Knowledge Distillation from Non-streaming to Streaming ASR Encoder using Auxiliary Non-streaming Layer. In *Proc. INTERSPEECH 2023*, pages 1663–1667.

Hubert Siuzdak, Piotr Dura, Pol van Rijn, and Nori Jacoby. 2022. WavThruVec: Latent speech representation as intermediate features for neural speech synthesis. In *Proc. Interspeech 2022*, pages 833–837.

Yun Tang, Anna Sun, Hirofumi Inaguma, Xinyue Chen, Ning Dong, Xutai Ma, Paden Tomasello, and Juan Pino. 2023. Hybrid transducer and attention based encoder-decoder modeling for speech-to-text tasks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12441–12455, Toronto, Canada. Association for Computational Linguistics.

Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. 2019. Transformer asr with contextual block processing. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 427–433. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020a. 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. Association for Computational Linguistics.

Chen Wang, Yuchen Liu, Boxing Chen, Jiajun Zhang, Wei Luo, Zhongqiang Huang, and Chengqing Zong. 2022. Discrete cross-modal alignment enables zero-shot speech translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5291–5302, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chengyi Wang, Yu Wu, Liang Lu, Shujie Liu, Jinyu Li, Guoli Ye, and Ming Zhou. 2020b. Low Latency End-to-End Streaming Speech Recognition with a Scout Network. In *Proc. Interspeech 2020*, pages 2112–2116.

Shushu Wang, Jing Wu, Kai Fan, Wei Luo, Jun Xiao, and Zhongqiang Huang. 2023. Better simultaneous translation with monotonic knowledge distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2334–2349, Toronto, Canada. Association for Computational Linguistics.

Yuxin Wu and Kaiming He. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Chen Xu, Bojie Hu, Yanyang Li, Yuhao Zhang, Shen Huang, Qi Ju, Tong Xiao, and Jingbo Zhu. 2021. Stacked acoustic-and-textual encoding: Integrating the pre-trained models into speech translation encoders. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2619–2630, Online. Association for Computational Linguistics.

Jichen Yang, Kai Fan, Minpeng Liao, Boxing Chen, and Zhongqiang Huang. 2023. Towards zero-shot learning for end-to-end cross-modal translation models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13078–13087, Singapore. Association for Computational Linguistics.

Xiaoyu Yang, Qiujia Li, and Philip C. Woodland. 2022. Knowledge distillation for neural transducers from large self-supervised pre-trained models. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8527–8531.

Rong Ye, Mingxuan Wang, and Lei Li. 2021. End-to-End Speech Translation via Cross-Modal Progressive Training. In *Proc. Interspeech 2021*, pages 2267–2271.

Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L Seltzer. 2019. Transformer-transducer: End-to-end speech recognition with self-attention. *arXiv preprint arXiv:1910.12977*.

Cheng Yi, Jianzhong Wang, Ning Cheng, Shiyu Zhou, and Bo Xu. 2021. Applying wav2vec2.0 to speech recognition in various low-resource languages. *arXiv preprint arXiv:2012.12121*.

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.

Linlin Zhang, Kai Fan, Jiajun Bu, and Zhongqiang Huang. 2023a. Training simultaneous speech translation with robust and random wait-k-tokens strategy. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7814–7831, Singapore. Association for Computational Linguistics.

Linlin Zhang, Kai Fan, Boxing Chen, and Luo Si. 2023b. A simple concatenation can effectively improve speech translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1793–1802, Toronto, Canada. Association for Computational Linguistics.

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

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

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

Shaolei Zhang and Yang Feng. 2023b. Unified segment-to-segment framework for simultaneous sequence generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shiliang Zhang, Zhifu Gao, Haoneng Luo, Ming Lei, Jie Gao, Zhijie Yan, and Lei Xie. 2020. Streaming Chunk-Aware Multihead Attention for Online End-to-End Speech Recognition. In *Proc. Interspeech 2020*, pages 2142–2146.

Libo Zhao, Kai Fan, Wei Luo, Wu Jing, Shushu Wang, Ziqian Zeng, and Zhongqiang Huang. 2023. Adaptive policy with wait-k model for simultaneous translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4832, Singapore. Association for Computational Linguistics.

A Data Statistics

We evaluate our model on MuST-C V1 English-German (EnDe) and English-Spanish (EnEs), and LibriSpeech 960h datasets. For ST data, following (Dong et al., 2022; Papi et al., 2023b; Fu et al., 2023a), we filter out short speech of less than 1000 frames (62.5ms) and long speech of more than 480,000 frames (30s) in the training set. The data statistics are illustrated in Table 3.

split	EnDe	EnEs	LibriSpeech	
train	225,271	260,041	281,242	
dev	1,418	1,312	2,703 (clean)	2,864 (other)
test	2641	2502	2620 (clean)	2939 (other)

Table 3: Number of samples for each split of finetuning data.

B The Number of Parameters of The Models

The number of parameters of wav2vec-S and the fine-tuned model are provided in Table 4 and 5.

		BASE	LARGE
CNN	strides	5, 2, 2, 2, 2, 2, 2	
	kernel width	10, 3, 3, 3, 3, 2, 2	
	channel	512	
Encoder	layer	12	24
	embedding dim.	768	1024
	inner FFN dim.	3072	4096
	attention heads	8	16
Projection	dim.	256	768
Number of Parameters		91M	309M

Table 4: Model architecture summary for wav2vec-S BASE and LARGE models

C Additional Training Details

For speech data, we use the raw 16-bit 16kHz mono-channel audio wave. For text data, the SentencePiece[14] model (Kudo and Richardson, 2018) is used to learn a unigram vocabulary of size 10k. We use an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$. For streaming ST finetuning, the learning rate is set to 2e-4 for the BASE and LARGE models. We warm up the learning rate for the first 4k steps and then decay it with the inverse square root schedule. For streaming ASR finetuning, we set learning rate to 5e-4 and 1e-4 for the BASE and LARGE models, respectively. Following wav2vec 2.0 (Baevski et al., 2020b), we use a tri-state rate schedule that initially warms up the learning rate during the first 10% of updates, maintains it at a steady level for the subsequent 30%, and then applies a linear decrease for the remaining updates, and wav2vec-S is not updated in the first 10k steps. The label smoothing and dropout rate are set to 0.1 for robust training. We train on 8 A100 GPUs with the max speech tokens of 1,400,000 (87.5 seconds) per GPU for BASE model, and the max speech tokens of 800,000 (50 seconds) per GPU for LARGE model. If the loss on the `dev` set does not decrease over ten epochs, the training process will be stopped early. Both models are trained for about 30h, where the LARGE model converges faster.

D Additional Results

D.1 Comparison with EDAtt

The baselines we compared in Section 4.2 do not leverage additional machine translation (MT) data, which can typically provide further improvement. EDAtt (Papi et al., 2023b), one of the strong baselines, utilizes an MT model trained on the OPUS dataset for knowledge distillation, making a direct comparison unfair. Therefore, we opted not to include a direct comparison with EDAtt in Section 4.2. Despite not utilizing additional MT data, our model still significantly outperforms EDAtt in low-latency regions, with improvements of about 6 BLEUs on both the EnDe and EnEs datasets, as indicated in the Figure 7. Moreover, Our model achieves comparable performance to EDATT in medium to high latency areas.

D.2 Latency-Quality Curves for Different Data Sizes

We present the latency-quality curves for different fine-tuned data sizes in Figures 8, 9, and 10.

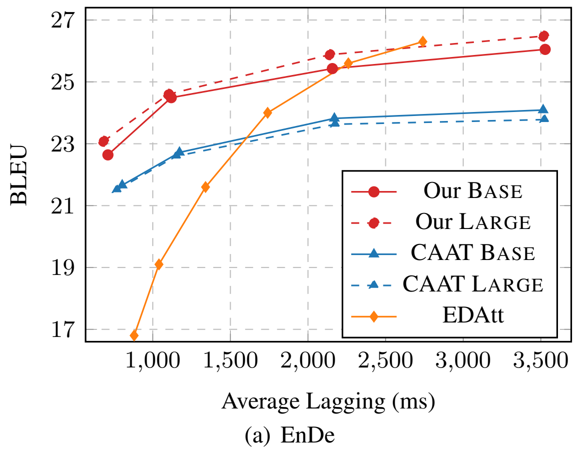
[14]`https://github.com/google/sentencepiece`

	encoder layer	decoder layer	embed dim	inner FFN dim	#Params (M)
Our BASE	12	6	768	3072	183
Our LARGE	24	12	1024	4096	622
CAAT BASE	12	6	768	3072	182
CAAT LARGE	24	12	1024	4096	620
MU-ST BASE	12 + 12	6	-	-	-
MU-ST LARGE	24 + 12	12	-	-	-
FAST	12 + 8	6	768	2048	202
MoSST	12 + 8	6	768	2048	202

Table 5: Model architecture summary for our finetuning and baseline models. MU-ST additionally contains a 12-layer Transformer layer for speech Segmentor. FAST and MoSST additionally include an 8-layer Transformer layer semantic encoder.

D.3 Numeric Results for the Figures

We also provide the numeric results for Figures 1 and 2 in Tables 6.



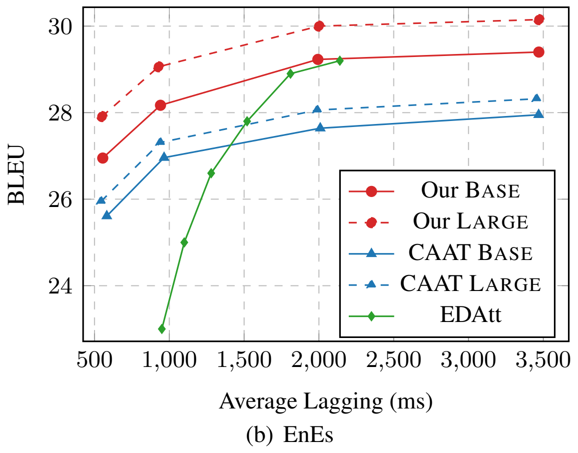


Figure 7: The translation quality (BLEU) against the latency metrics (AL) on the tst-COMMON set of MuST-C EnDe and EnEs datasets.

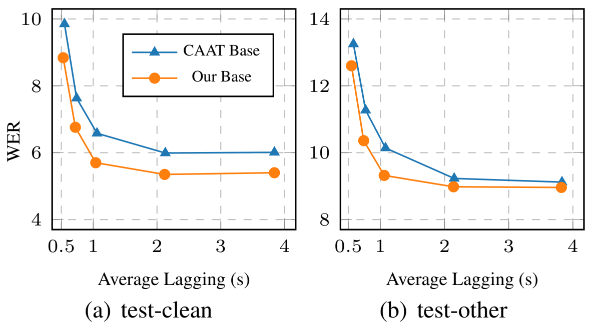


Figure 8: ASR quality (WER) vs. latency metrics (AL) on LibriSpeech 500h subset.

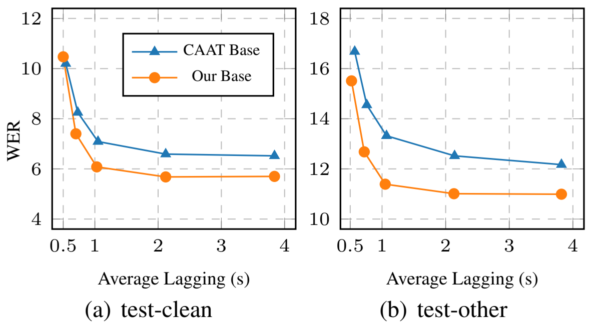


Figure 9: ASR quality (WER) vs. latency metrics (AL) on LibriSpeech 360h subset.

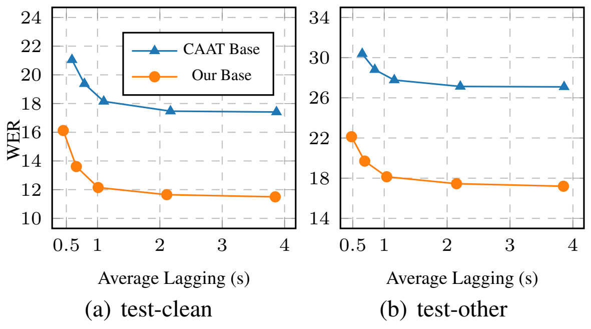


Figure 10: ASR quality (WER) vs. latency metrics (AL) on LibriSpeech 100h subset.

Model	d	En-De				En-Es				LibriSpeech test-clean				LibriSpeech test-other			
		AL	AP	DAL	BLEU	AL	AP	DAL	BLEU	AL	AP	DAL	WER	AL	AP	DAL	WER
Our Large	2	683	0.68	1327	23.08	550	0.67	1340	27.91	644	0.60	1205	6.37	678	0.61	1248	7.84
	4	1101	0.74	1799	24.60	928	0.73	1796	29.06	978	0.65	1704	4.69	1000	0.66	1715	6.36
	10	2140	0.86	3052	25.88	2000	0.85	3111	30.00	2086	0.79	3332	3.81	2108	0.80	3314	5.54
	20	3518	0.95	4385	26.48	3471	0.94	4530	30.15	3823	0.91	5164	3.52	3810	0.92	5042	5.26
Our Base	2	711	0.68	1336	22.64	555	0.67	1339	26.95	701	0.61	1258	6.11	728	0.62	1300	9.29
	4	1119	0.74	1795	24.49	941	0.73	1800	28.17	1024	0.66	1737	4.57	1043	0.67	1749	7.88
	10	2157	0.86	3056	25.43	1993	0.85	3101	29.23	2103	0.79	3345	3.93	2122	0.81	3326	7.22
	20	3528	0.95	4385	26.05	3470	0.94	4523	29.40	3826	0.91	5169	3.90	3813	0.92	5046	7.11
CAAT Large	2	769	0.69	1363	21.51	544	0.67	1334	25.95	753	0.62	1322	6.29	793	0.63	1375	9.62
	4	1152	0.74	1812	22.60	940	0.73	1797	27.31	1070	0.67	1776	5.20	1103	0.68	1802	8.43
	10	2175	0.86	3055	23.63	1989	0.85	3102	28.06	2134	0.80	3350	4.66	2158	0.81	3334	7.71
	20	3523	0.95	4375	23.78	3455	0.94	4522	28.32	3845	0.91	5167	4.50	3834	0.92	5045	7.47
CAAT Base	2	803	0.69	1398	21.66	582	0.68	1362	25.61	705	0.61	1270	6.19	735	0.62	1317	9.17
	4	1172	0.75	1826	22.72	965	0.73	1812	26.96	1031	0.66	1744	4.88	1051	0.67	1760	7.82
	10	2170	0.86	3062	23.82	2009	0.85	3105	27.64	2107	0.79	3340	4.25	2124	0.81	3322	7.00
	20	3515	0.95	4379	24.09	3469	0.94	4523	27.95	3829	0.91	5166	3.96	3814	0.92	5044	6.68

Table 6: Numeric results on MuST-C EnDe and EnEs `tst-COMMON` sets, and LibriSpeech `test-clean` and `test-other` sets for our models and CAAT models (Figure 1).