**Abstract**

End-to-end simultaneous speech translation (SST), which directly translates speech in one language into text in another language in real-time, is useful in many scenarios but has not been fully investigated. In this work, we propose RealTranS, an end-to-end model for SST. To bridge the modality gap between speech and text, RealTranS gradually downsamples the input speech with interleaved convolution and unidirectional Transformer layers for acoustic modeling, and then maps speech features into text space with a weighted-shrinking operation and a semantic encoder. Besides, to improve the model performance in simultaneous scenarios, we propose a blank penalty to enhance the shrinking quality and a Wait-K-Stride-N strategy to allow local reranking during decoding. Experiments on public and widely-used datasets show that RealTranS with the Wait-K-Stride-N strategy outperforms prior end-to-end models as well as cascaded models in diverse latency settings.

1 Introduction

Simultaneous speech translation (SST) (Fügen et al., 2007; Oda et al., 2014; Ren et al., 2020) aims to translate speech in one language into text in another language concurrently. It is useful in many scenarios, like synchronous interpretation in international conferences, automatic caption for live videos, etc. However, prior studies either focus on full sentence speech translation (ST) (Berard et al., 2016; Weiss et al., 2017; Liu et al., 2019) or simultaneous text-to-text machine translation (STT) (Cho and Esipova, 2016; Gu et al., 2017; Dalvi et al., 2018) which takes a segmented output from an automatic speech recognition (ASR) system as input. Such two-stage models (i.e., cascaded models) inevitably introduce error propagation and also increase translation latency (see Figure 1). Ren et al. (2020) propose an end-to-end SST system called SimulSpeech, but they ignore the modality gap between speech and text, which is important for improving translation quality (Liu et al., 2020).

In this paper, we propose RealTranS model for SST. To relieve the burden of our encoder (Wang et al., 2020c; Liu et al., 2020), we decouple it into three parts: acoustic encoder, weighted-shrinking operation, and semantic encoder. We apply Conv-Transformer (Huang et al., 2020) as our acoustic encoder, which gradually downsamples the input speech and learns acoustic information with interleaved convolution and Transformer layers. The weighted-shrinking operation bridges the length gap between speech and text, by weighted summing up the frames in one detected segment based on the posterior probabilities generated by a CTC module (Graves et al., 2006). Finally, we use a semantic encoder to extract semantic features and deliver them to the decoder for translation.
To enable simultaneous decoding, unidirectional Transformer is applied in our encoder. This inevitably affects the performance of the CTC module and the following shrinking operation. To alleviate this, we introduce a blank-limited CTC loss, which adds a blank penalty to the traditional CTC loss to encourage the model to produce non-blank labels, given the observation that CTC tends to produce peaky distribution as a kind of overfitting (Liu et al., 2018) by over predicting blank labels. Accordingly, the shrinking quality can be improved. Furthermore, we propose a new simultaneous strategy Wait-K-Stride-N which allows local reranking during decoding. This strategy can resolve the inherent drawback of the conventional Wait-K strategy (Ma et al., 2019), which cannot apply vanilla beam search efficiently (Zheng et al., 2019c).

Experiments on Augmented LibriSpeech En–Fr, MUST-C En–Es and En–De datasets demonstrate the effectiveness of the Wait-K-Stride-N strategy, and show that RealTranS achieves better performance than the prior end-to-end model SimulSpeech (Ren et al., 2020) as well as the cascaded models. Further analysis and ablation study reveal the effects of our proposed modules in RealTranS. We also compare RealTranS with other methods on full sentence ST. Results show that RealTranS achieves competitive or even better results, indicating its superiority.

In summary, the contributions of this work include the following aspects:

- We propose RealTranS for SST, which can gradually bridge the modality gap between speech and text with the help of gradual downsampling and weighted shrinking.
- We introduce a blank penalty and the Wait-K-Stride-N strategy to improve the performance in simultaneous translation scenarios.
- Extensive experiments on public and widely-used datasets show the superiority of our RealTranS model and our Wait-K-Stride-N strategy in diverse latency settings.

2 Related Work

Speech Translation. Speech translation (ST) has recently attracted intensive attention from the AI community. Earlier works are mostly based on cascaded models, which perform NMT on the outputs of ASR systems (Ney, 1999; Mathias and Byrne, 2006; Sperber et al., 2017; Bahar et al., 2021). Cascaded models inevitably introduce error propagation from ASR (Weiss et al., 2017). To avoid this problem and for better efficiency, end-to-end ST models are proposed and become popular in recent years (Berard et al., 2016, 2018; Bansal et al., 2018; Gangi et al., 2019). To alleviate the data scarcity problem of end-to-end ST models, various techniques are utilized, including pre-training (Bansal et al., 2019), multi-task learning (Anastasopoulos and Chiang, 2018), knowledge distillation (Liu et al., 2019; Ren et al., 2020), data synthesis (Jia et al., 2019), self-supervised learning (Chen et al., 2020) and speech augmentation techniques like SpecAugment (Bahar et al., 2019) or speed perturbation (Stoian et al., 2020).

Some studies focus on how to bridge the gap between different modalities (speech and text) or different modules (acoustic and semantic modeling). Wang et al. (2020b,c) propose a TCEN model and a curriculum pre-training technique to make sure the modules learn desired information, respectively. Salesky and Black (2020) explore phone features as intermediate representations to improve performance, while Dong et al. (2020b) use pre-trained BERT to guide the model to learn semantic knowledge. Modality Agnostic Meta-Leaning is also exploited for ST in Indurthi et al. (2019). To bridge the length gap, Zhang et al. (2020) propose adaptive feature selection, while Dong et al. (2020a) and Liu et al. (2020) exploit the CTC-based (Graves et al., 2006) shrinking mechanism. Nevertheless, they do not explore in simultaneous scenarios, where encoding quality inevitably suffers because of lacking future information in unidirectional encoders.

Simultaneous Translation. Previous studies on simultaneous translation focus on text-to-text scenarios (STT) (Cho and Esipova, 2016; Gu et al., 2017; Dalvi et al., 2018), where fixed policies (Ma et al., 2019) and adaptive policies (Arivazhagan et al., 2019; Zheng et al., 2019b,a) are proposed to decide when to read and write tokens. (Ma et al., 2019) proposed a simple yet effective strategy, Wait-K, based on a prefix-to-prefix framework. It first waits for the first k tokens, and then start to generate target tokens concurrently with the source stream. It achieves competitive performance in simultaneous translation (Zheng et al., 2019a, 2020).

Traditional simultaneous speech-to-text translation (SST) mainly depends on the ASR segmentation and then performs NMT based on the stream-
ing segmented chunks (Oda et al., 2014; Iranzo-Sánchez et al., 2020). There is little attention on end-to-end SST. Ren et al. (2020), to our knowledge, first propose an end-to-end model called SimulSpeech with multi-task learning and knowledge distillation, and apply the Wait-K strategy to perform simultaneous translation. Ma et al. (2020a) explore how to define a “token” in source speech, and then adapt methods from STT to SST. And Ma et al. (2020b) introduce a memory-augmented Transformer to tackle the streaming speech input. However, none of them investigate the modality gap between speech and text.

3 The RealTranS Model

Our RealTranS follows the sequence-to-sequence architecture, which consists of an ST encoder and an ST decoder. The ST encoder is decoupled into three parts, an acoustic encoder, a weighted shrinking operation, and a semantic encoder, to gradually map speech inputs into semantic representation space of text. Figure 2 shows the architecture.

3.1 Problem Formulation

Speech translation corpora usually contain triples of speech, transcription and translation, denoted as \( D_{ST} = \{(x, z, y)\} \). Specifically, \( x = (x_1, x_2, \ldots, x_{T_x}) \) is a sequence of speech features extracted from speech signals, e.g., filterbanks. \( z = (z_1, z_2, \ldots, z_{T_z}) \) and \( y = (y_1, y_2, \ldots, y_{T_y}) \) are the corresponding transcription in source language and translation in target language. \( T_x, T_z, \) and \( T_y \) are the lengths of speech, transcription and translation, respectively, where usually \( T_x \gg T_z \) and \( T_x \gg T_y \). A typical end-to-end model only makes use of \( x \) and \( y \), while \( z \) can be used as multi-task training for other objectives, like CTC loss.

3.2 Acoustic Encoder

Acoustic encoder mainly encodes speech features \( x \) into a hidden space to learn acoustic knowledge. We apply Conv-Transformer (Huang et al., 2020) to extract the desired features. It contains three blocks, each of which is composed of three convolution layers followed by unidirectional Transformer layers (see lower left in Figure 2) to prevent from leveraging future context in SST. Similar to Huang et al. (2020), we make the model aware of limited future frames with a look-ahead window in the convolution layers, to help improve acoustic modeling. At the same time, we gradually downsample the long speech features by setting the stride size to 2 in the second convolution layer in each block. In this way, the speech features are gradually reduced and approach the length of the corresponding text.

To predict the word boundaries\(^1\) in the speech input and further improve the learned acoustic features, we adopt the Connectionist Temporal Classification (CTC) (Graves et al., 2006) module on top of the acoustic encoder. This module contains a Multilayer Perceptron (MLP) followed by a Softmax operator. CTC predicts a path \( \pi = (\pi_1, \pi_2, \ldots, \pi_{T'_x}) \), where \( T'_x \) is the length of hidden states after the Conv-Transformer. And \( \pi_t \in V \cup \{\phi\} \) can be either a token in the source

\(^1\)For ease of understanding, we use word boundaries here. Same methods can be applied to boundaries between phones, chars, subwords, etc., depending on the granularity of the CTC loss units. We use subword units in our experiments.
vocabulary $\mathcal{V}$ or the blank symbol $\phi$. CTC paths have the many-to-one mapping to output sequences by removing blank symbols and consecutively repeated labels, denoted as operation $B$. Therefore, CTC loss is defined as follows:

$$L_{CTC} = - \sum_{(x,z) \in D} \sum_{\pi \in B^{-1}(z)} p(\pi | x) \quad (1)$$

where $B^{-1}(z)$ denotes all possible CTC paths that can be mapped to the transcription $z$. With the CTC module, we can define a word boundary between two frames where the first frame has a non-blank label while the second frame has a different label from the first one. Figure 3 shows an example of word boundaries.

Due to the data scarcity problem in ST and CTC’s inherent characteristics (Liu et al., 2018), the detected word boundaries are usually not accurate enough. The module will overly predict the occurrence of blank labels (as a kind of overfitting), resulting in a large gap between the number of detected boundaries and the number of tokens in transcription, especially when unidirectional Transformer is applied (see Table 3). To alleviate the problem, we add a blank penalty to encourage the module to produce non-blank labels. It is called blank-limited CTC loss and defined as follows:

$$L_{CTC} = L_{CTC} + \lambda \sum_{x \in D} \sum_{\pi \in \pi(x)} p(\pi | x) \quad (2)$$

where $\pi(x)$ means the argmax results of CTC softmax outputs. $\lambda$ controls the effect of blank penalty.

### 3.3 Weighted-Shrinking Operation

The length gap between acoustic features and the corresponding transcription and translation is still large after the gradual downsampling in Section 3.2. Inspired by previous studies (Dong et al., 2020a; Liu et al., 2020), we adopt a shrinking operation to bridge the gap based on CTC predictions. Prior works usually either remove blank frames and then average repeated frames (Dong et al., 2020a) or select a single representative frame (Liu et al., 2020) in a detected segment (i.e., frames between two word boundaries). However, there might be useful information in the detected blank or repeated frames, especially when the boundary detection is not accurate enough (see Section 3.2).

We propose a weighted-shrinking mechanism to tackle the problem. We assume that the probability of a frame to be labeled as “blank” represents the confidence that the model “thinks” it is not important. Therefore, for the frames in one segment, their weights are decided by the probabilities to be blank labels. The representation of the segment will be the weighted average of the corresponding frames. The specific operation is shown as follows:

$$h_t' = \sum_{l \in s(t)} h_l \frac{\exp(\mu (1 - p_l^b))}{\sum_{s \in s(t)} \exp(\mu (1 - p_s^b))} \quad (3)$$

where $p_l^b$ denotes the probability of the frame $t$ to be blank, and $h_t$ represents the hidden state of the frame $t$ in our acoustic encoder. $\mu \geq 0$ controls the temperature of the distribution (i.e., Softmax function). When $\mu = 0$, it means that we simply average the frames; and when $\mu \to \infty$ it degenerates to the general shrinking mechanism where only the representative frames with the highest confidence are selected.

### 3.4 Semantic Encoder

The shrinking operation only bridges the length gap between speech and text, but the shrunk representations still lack semantic information. Therefore, we apply a semantic encoder on top of the shrunk representations (Liu et al., 2020). It first applies a positional embedding layer, and then follows several Transformer layers (also unidirectional to mask future context), to extract semantic representations.

### 3.5 ST Decoder

A similar decoder as the basic Transformer architecture in NMT is adopted, where several Transformer decoder layers are stacked on top of target embeddings. To simulate simultaneous translation, we follow the prefix-to-prefix framework (Ma et al., 2019) and mask certain future context in cross attention to ensure that the model predicts the current token based on only part of the input from the ST encoder. How much context the model can see depends on the simultaneous strategy that is applied. For example, a Wait-K (Ma et al., 2019) based decoder predicts the $t$-th token based on the first $t + k - 1$ hidden states produced by the ST encoder.

**Wait-K-Stride-N.** There is one drawback in the conventional Wait-K strategy – it cannot perform vanilla beam search while decoding except for the
Figure 4: Wait-K vs. Wait-K-Stride-N (in this example, we set $k = 3$ and $n = 2$).

long-tail scenario (Ma et al., 2019; Zheng et al., 2019c), though beam search has been proven very effective in improving translation quality. Based on the prefix-to-prefix framework and STATIC-RW strategy proposed in Dalvi et al. (2018), we propose the Wait-K-Stride-N strategy to allow using beam search for local reranking during simultaneous decoding. Similar to the Wait-K strategy, our strategy first reads $k$ input units (tokens in MT or segments in ST). Then, the model repeatedly performs $n$ write and read operations until the end of the sentence (see Figure 4). In this way, the translation latency is close to Wait-K, but we can perform beam search during the $n$ write operations. The objective with such a strategy is hence defined as follows:

$$
L_{ST} = -\sum_{(x,y) \in D} \prod_{t=1}^{T_y} p(y_t | y_{<t}, x_{\leq t})^{\lfloor (t-1)/n \rfloor + k}
$$

(4)

where $y_{<t}$ represents the target tokens before $y_t$, $T_y$ is the length of the target sentence, and $x_{\leq t}$ represents the first $t$ detected source segments.

3.6 Training Procedure
The total objective of our model will be the sum of the CTC part and the ST part:

$$
\mathcal{L} = \mathcal{L}_{CTC} + \alpha \mathcal{L}_{CTC} \quad (5)
$$

where $\alpha$ controls the influence of the CTC part.

To enhance the CTC quality, we also apply a pre-training procedure (Stoian et al., 2020). We only use CTC loss to pre-train the acoustic encoder\(^2\). In this way, we can prevent the training waste (Wang et al., 2020b), and focus on improving the alignment results, which are essential for shrinking operations (see Table 3). The whole model is then fine-tuned with the whole ST corpus.

\(^2\)We do not pre-train the decoder for simplicity though it might further improve our performance.

| Corpus  | Train  | Dev    | Test   |
|---------|--------|--------|--------|
| En–Fr   | 47,271 × 2 (100h) | 1,071 (2.0h) | 2,048 (4.0h) |
| En–Es   | 265,625 (496h)     | 1,316 (2.5h) | 2,502 (4.0h) |
| En–De   | 229,703 (400h)     | 1,423 (2.5h) | 2,641 (4.0h) |

Table 1: The number of sentences and the duration of audios for Augmented LibriSpeech En–Fr, MuST-C En–Es and En–De datasets.

4 Experimental Setup

4.1 Datasets
We conduct experiments on three publicly available datasets: Augmented LibriSpeech English-French (En–Fr) corpus (Kocabiyikoglu et al., 2018), and MUST-C English-Spanish (En–Es) and English-German (En–De) corpus (Di Gangi et al., 2019). All the datasets include source audios with the corresponding transcriptions in source language and translations in target language. For the Augmented Librispeech En–Fr corpus, we follow previous work (Wang et al., 2020c) and use the 100-hour clean training set with aligned references and provided Google translations, resulting in double size of training pairs. For MUST-C datasets, We use the official data splits for train and development and tst-COMMON set for test. The statistics for these three datasets are listed in Table 1.

4.2 Experimental Settings
We use 80-dimensional log-mel filterbanks as acoustic features, which are calculated with 25ms window size and 10ms step size and normalized by utterance-level Cepstral Mean and Variance Normalization (CMVN). For transcriptions and translations, SentencePiece\(^3\) (Kudo and Richardson, 2018) is used to generate subword vocabularies with the sizes of 4k and 8k respectively. We remove the punctuation in transcriptions.

Our acoustic encoder follows the settings of the original Conv-Transformer (Huang et al., 2020), except that the channel number in convolution layers and hidden size and head number in Transformer layers are half values of theirs. This means the output dimension of the acoustic encoder is 256. We use 6 Transformer layers in the semantic encoder and 4 in the ST decoder. The hyper-parameters $\lambda$ (Eq. 2), $\mu$ (Eq. 3) and $\alpha$ (Eq. 5) in our model are set to 0.5, 1.0 and 1.0, respectively.

\(^3\)https://github.com/google/sentencepiece
Our model is trained with 8 NVIDIA Tesla V100 GPUs, batched with an approximate 40000-frame features. We use Adam optimizer (Kingma and Ba, 2015) with a 0.002 learning rate and 10000 warm-up steps followed by the inverse square root scheduler. Dropout strategy is used with a rate of 0.1. We save checkpoints every epoch and average the last 10 checkpoints for evaluation with a beam size of 5. For simplicity, we use the same K and N values as those of training for inference. We implement our model based on Fairseq S2T (Wang et al., 2020a).

4.3 Evaluation Metrics

We apply SacreBLEU for translation quality evaluation unless otherwise stated. For the metrics of latency, we adapt Average Proportion (AP) (Cho and Esipova, 2016) and Average Lagging (AL) (Ma et al., 2019) to ST settings, following previous studies (Ren et al., 2020; Ma et al., 2020a).

**Average Proportion.** AP calculates the mean absolute latency cost by each target token, where we replace the steps of source tokens with the time spent. It can be calculated as follows:

\[
AP(x, y) = \frac{1}{|x||y|} \sum_{i=1}^{w} d(y_i)
\]

where \(d(y_i)\) is the speech duration that has been listened when producing the target token \(y_i\).

**Average Lagging.** AL evaluates the degree of that the user is out of sync with the speaker, in terms of the number of source tokens (Ma et al., 2019). Following Ma et al. (2020a), we also extend it to the basis of time duration rather than source tokens, which is defined as follows:

\[
AL(x, y) = \frac{1}{\tau(|x|)} \sum_{i=1}^{\tau(|x|)} [d(y_i) - \frac{|x|}{|y^*|} T_s (i - 1)]
\]

where \(|y^*|\) is the length of the reference translation, and \(\tau(|x|)\) denotes the index of the corresponding target token when our model has read the entire source speech. \(T_s\) represents that the speech features are extracted every \(T_s\) ms (decided by the step size in the feature extraction and downsampling rates in convolution layers), which will be 80ms in our model. What’s more, our Conv-Transformer module introduces a 140ms look-ahead window (Huang et al., 2020), so we add 140 to the final AL scores to be fairly compared with other models.

https://github.com/pytorch/fairseq/tree/master/examples/speech_to_text

https://github.com/mjpost/sacreBLEU

5 Experimental Results

This section displays our experimental results. To explicitly show the performance trend of models in different scenarios, we use line charts to display
most of the compared results. Their corresponding numeric results can be found in Appendix A.

5.1 Translation Quality vs. Latency

We first evaluate our RealTranS model with our Wait-K-Stride-N simultaneous strategy on the three datasets. We select three values 1, 2 and 3 for N to compare (it becomes conventional Wait-K when N=1). The results are displayed in Figure 5.

Results show that RealTranS achieves higher BLEU scores as the K value increases, with sacrifice of translation delay, consistent with prior works (Ren et al., 2020; Ma et al., 2020a). Compared to the conventional Wait-K (N=1), our model with N=2 can achieve better BLEU scores under the same latency requirements, which demonstrates the effectiveness of our proposed Wait-K-Stride-N strategy. When N=3, the latency becomes higher. And it only achieves similar gains in BLEU scores compared to N=2 on MUST-C En–Es and En–De datasets. Therefore, we will use N=2 as our simultaneous strategy in later experiments unless otherwise stated.

5.2 Comparison with SimulSpeech

We compare with SimulSpeech (Ren et al., 2020), the state-of-the-art end-to-end model for SST. Figure 6 shows the performance comparison on the MUST-C En–Es dataset (we report tokenized case-sensitive BLEU scores following their settings). Since they only report segment-based AL, we transfer it to our time-based AL proportionally based on the latency when K=inf.

We find that our RealTranS model outperforms SimulSpeech almost in all latency settings, with an average of about 3 higher BLEU scores. Although SimulSpeech achieves relatively lower latency (e.g., less than 1000 ms AL when K=1), the performance inevitably suffers. What’s more, SimulSpeech has leveraged multi-task learning and knowledge distillation to enhance their performance, which can be also applied to further improve the performance of our RealTranS model.

5.3 Comparison with Cascaded Model

We implement a cascaded model to compare with RealTranS under the same latency. Specifically, we combine our acoustic encoder (Section 3.2) and a Transformer decoder as our ASR model and use the conventional Transformer encoder-decoder architecture as our NMT model. Their configuration is similar to RealTranS (e.g., the same hidden dimension and the same number of Transformer layers). And we train the ASR and NMT model with the same corpus. The conventional Wait-K strategy is used in the ASR model since the alignment between speech and transcription is monotonic, while Wait-K-Stride-N is applied in the NMT model. Since several combinations of the ASR and NMT models may be under the same latency, we report the best BLEU score among them.

Table 2 shows the comparison results on MUST-C En–Es dataset. We have the following observations: 1) our RealTranS model outperforms the cascaded model in all latency settings, which demonstrates the superiority of RealTranS. 2) The improvement over the cascaded model becomes larger when the value of K is smaller. This observation is consistent with Ren et al. (2020). We attribute this to the advantage of end-to-end models over cascaded models, where the impact of error propagation in cascaded models may be amplified when the latency is low.

| Model      | K=2  | K=4  | K=6  | K=8  | K=10 | K=inf |
|------------|------|------|------|------|------|-------|
| Cascaded   | 14.92| 19.22| 22.11| 23.33| 24.47| 26.79 |
| RealTranS  | 18.45| 22.65| 24.79| 25.41| 25.82| 27.40 |

Table 2: Comparison with the cascaded model using Wait-K-Stride-N strategy when N=2 in terms of BLEU scores. The same K value means similar latency.
Table 3: Shrinking quality of RealTranS. “CTC PT” indicates that the acoustic encoder is pre-trained with CTC. “BP” represents blank penalty, and “Bi-Enc” means that using bidirectional Transformer encoders. “Diff ≤ n” means the difference between the length of shrunk representations and that of the ground-truth transcription is less than or equal to n. We report the percentage of the cases on the MUST-C En–Es test set. The BLEU scores displayed are results when K=inf.

5.4 Effects of Blank Penalty and Weighted Shrinking Operation

In this subsection, we examine the effects of our proposed methods, including blank penalty (Eq. 2) and weighted-shrinking operation (Eq. 3).

Blank Penalty. We propose a blank penalty to alleviate the inaccuracy of alignments between speech features and transcriptions when applying unidirectional encoders for SST (Section 3.2). To examine the effect, we evaluate the shrinking quality, i.e., the differences between the length of representations after the shrinking operation and that of the ground-truth transcription, on MUST-C En–Es dataset, and display the statistics in Table 3. We can see that the performance, as well as the shrinking quality, drops when removing CTC pre-training. It further decreases when removing the blank penalty, which shows its effectiveness. Also, we can see that the performance loss partly comes from using unidirectional encoders rather than bidirectional for simultaneous purposes (the 4th row), which can be compensated by the blank penalty.

Weighted-Shrinking. To validate our weighted-shrinking operation, we first investigate the effects of various values for the shrinking temperature \( \mu \) in Eq. 3 and display the results in Figure 7(a). The results show that our weighted-shrinking mechanism (\( \mu = 1.0 \)) performs better than both simply averaging all the frames (\( \mu = 0 \)) and dropping blank frames (“DB”).

We also try to replace our weighted-shrinking module with another Conv-Transformer block (see Section 3.2), resulting in a model with 240ms downsample rate in total (denoted as “4 blocks w/o shrink”). Figure 7(b) shows comparison results, together with a model with only 2 blocks (40ms downsample rate, denoted as “2 blocks with shrink”). We can find that downsampling only with convolution layers performs worse than RealTranS, while less downsample rate also affects the performance. This implies that there is an upper-bound for downsampling with only convolution layers while maintaining performance, and our weighted-shrinking operation can be used as an addition to further improve the performance.

5.5 Ablation Study

We evaluate the contributions of different modules in RealTranS. Each module is evaluated in four kinds of latency settings: Wait-2-Stride-2, Wait-6-Stride-2, Wait-10-Stride-2, and also Wait-Inf (full sentence translation). The results are shown in Figure 8, where “-CTC PT” means we do not pre-train the encoder with CTC loss and “-BP” indicates we further remove the blank penalty. “-shrink” means removing the weighted-shrinking operation and the semantic encoder, while “-GD” denotes disabling the gradual downsampling by moving the Trans-
former layers in block 1 & 2 to block 3 (see Figure 2) to perform downsampling at the beginning layers of the acoustic encoder. Finally, “-CTC” indicates that CTC objective is removed.

Figure 8 shows that all modules play a role in RealTranS. Specifically, we have the following observations: 1) The CTC module is important for improving translation quality, and it can be further improved by pre-training. 2) The blank penalty is useful in reducing latency while maintaining translation quality. 3) The blank penalty is essential for the shrinking operation since on full sentence translation the shrinking operation degrades the performance without the blank penalty (“-BP” vs “-Shrink”). 4) Gradual downsampling also contributes to the performance, because directly downsampling to a large rate may make it difficult to learn acoustic features.

5.6 Comparison in Full Sentence Translation

Although focusing on SST, RealTranS can also be applied in full sentence ST. For fair comparison, we replace the unidirectional Transformer layers in RealTranS with bidirectional ones and report both case-insensitive and case-sensitive BLEU scores following prior works. Table 4 displays the BLEU scores compared with existing methods (we only compare with end-to-end models trained with the same data for fair comparison) on Augmented LibriSpeech En–Fr and MUST-C En–De datasets. Results marked with * are case-sensitive BLEU scores, while the others are case-insensitive.

6 Conclusion

This work proposes a new end-to-end model RealTranS and a new strategy Wait-K-Stride-N for SST. RealTranS gradually bridges the modality gap between speech and text, and achieves new state-of-the-art results for SST. Empirical studies have shown the proposed blank penalty for CTC loss helps on the alignment with transcription, which reduces latency while maintaining translation quality. Our weighted-shrinking operation, as well as Wait-K-Stride-N simultaneous strategy, further improves the performance. We also compare RealTranS with other methods for full sentence translation, where RealTranS still exhibits competitive results, showing its superiority.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 82–91, New Orleans, Louisiana. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019.
Monotonic infinite lookback attention for simultaneous machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Parnia Bahar, Tobias Bieschke, Ralf Schlüter, and Hermann Ney. 2021. Tight integrated end-to-end training for cascaded speech translation. In IEEE Spoken Language Technology Workshop, SLT 2021, Shenzhen, China, January 19-22, 2021, pages 950–957. IEEE.

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On using specAugment for end-to-end speech translation. CoRR, abs/1911.08876.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2018. Low-resource speech-to-text translation. In Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018, pages 1298–1302. ISCA.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. 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 58–68, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexandre Berard, Laurent Besacier, Ali Can Koçabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018, pages 6224–6228. IEEE.

Alexandre Berard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. CoRR, abs/1612.01744.

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

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.

Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2020a. Consecutive decoding for speech-to-text translation. CoRR, abs/2009.09737.

Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2020b. Listen, understand and translate: Triple supervision decouples end-to-end speech-to-text translation. CoRR, abs/2009.07904.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. Mach. Transl., 21(4):209–252.

Mattia Antonino Di Gangi, Matteo Negri, and Marco Turchi. 2019. Adapting transformer to end-to-end spoken language translation. In Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019, pages 1133–1137. ISCA.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006, volume 148 of ACM International Conference Proceeding Series, pages 369–376. ACM.

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

Wenyong Huang, Wenchao Hu, Yu Ting Yeung, and Xiao Chen. 2020. Conv-transformer transducer: Low latency, low frame rate, streamable end-to-end speech recognition. In Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020, pages 3001–3005. ISCA.

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.
Elizabeth Salesky and Alan W Black. 2020. Phone features improve speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2388–2397, Online. Association for Computational Linguistics.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1380–1389, Copenhagen, Denmark. Association for Computational Linguistics.

Mihaela C. Stoian, Sameer Bansal, and Sharon Goldwater. 2020. Analyzing ASR pretraining for low-resource speech-to-text translation. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 7909–7913. IEEE.

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.

Chengyi Wang, Yu Wu, Shujie Liu, Zhenglu Yang, and Ming Zhou. 2020b. Bridging the gap between pre-training and fine-tuning for end-to-end speech translation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9161–9168. AAAI Press.

Chengyi Wang, Yu Wu, Shujie Liu, Ming Zhou, and Zhenglu Yang. 2020c. Curriculum pre-training for end-to-end speech translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3728–3738, Online. Association for Computational Linguistics.

Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. 2017. Sequence-to-sequence models can directly transcribe foreign speech. *CoRR*, abs/1703.08581.

Biao Zhang, Ivan Titov, Barry Haddow, and Rico Sennrich. 2020. Adaptive feature selection for end-to-end speech translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2533–2544, Online. Association for Computational Linguistics.
## Appendix

### A Numeric Results for the Figures

| N Value | K=N | K=N+2 | K=N+4 | K=N+6 | K=N+8 |
|---------|-----|-------|-------|-------|-------|
|         | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    |
| N=1     | 12.72 | 1710 | 15.04 | 2263 | 15.9 | 2748 | 16.6 | 3160 | 16.71 | 3572 |
| N=2     | 13.51 | 1792 | 15.63 | 2378 | 16.37 | 2856 | 16.87 | 3291 | 17.18 | 3669 |
| N=3     | 14.81 | 2004 | 16.13 | 2519 | 16.84 | 2962 | 17.04 | 3383 | 17.26 | 3746 |

Table 5: Numeric Results for Figure 5(a).

| N Value | K=N | K=N+2 | K=N+4 | K=N+6 | K=N+8 |
|---------|-----|-------|-------|-------|-------|
|         | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    |
| N=1     | 16.32 | 916 | 21.23 | 1426 | 23.53 | 1924 | 25.06 | 2399 | 25.45 | 2830 |
| N=2     | 18.45 | 1047 | 22.65 | 1554 | 24.79 | 2043 | 25.41 | 2514 | 25.82 | 2920 |
| N=3     | 21.74 | 1445 | 24.26 | 1968 | 25.11 | 2461 | 25.29 | 2944 | 25.78 | 3356 |

Table 6: Numeric Results for Figure 5(b).

| N Value | K=N | K=N+2 | K=N+4 | K=N+6 | K=N+8 |
|---------|-----|-------|-------|-------|-------|
|         | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    |
| N=1     | 13.22 | 1100 | 16.65 | 1582 | 18.67 | 2059 | 19.66 | 2508 | 19.78 | 2913 |
| N=2     | 15.74 | 1233 | 18.12 | 1709 | 19.02 | 2169 | 20.06 | 2607 | 20.09 | 3005 |
| N=3     | 16.54 | 1355 | 18.49 | 1838 | 19.84 | 2290 | 20.05 | 2720 | 20.41 | 3106 |

Table 7: Numeric Results for Figure 5(c).

| Model    | K=N | K=N+2 | K=N+4 | K=N+6 | K=N+8 | K=inf |
|----------|-----|-------|-------|-------|-------|-------|
|          | BLEU | AP    | BLEU | AP    | BLEU | AP    | BLEU | AP    | BLEU | AP    |
| SimulSpeech | 15.02 | 0.550 | 19.92 | 0.700 | 21.58 | 0.785 | 22.42 | 0.840 | 22.49 | 0.885 | 22.72 | 1.0 |
| RealTranS(N=2) | 18.54 | 0.654 | 22.74 | 0.730 | 24.89 | 0.793 | 25.54 | 0.842 | 25.97 | 0.877 | 27.54 | 1.0 |

Table 8: Numeric Results for Figure 6(a).

| Model    | K=N | K=N+2 | K=N+4 | K=N+6 | K=N+8 | K=inf |
|----------|-----|-------|-------|-------|-------|-------|
|          | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    | BLEU | AL    |
| SimulSpeech | 15.02 | 694 | 19.92 | 1336 | 21.58 | 2169 | 22.42 | 2724 | 22.49 | 3331 | 22.72 | 6141 |
| RealTranS(N=2) | 18.54 | 1047 | 22.74 | 1554 | 24.89 | 2043 | 25.54 | 2514 | 25.97 | 2920 | 27.54 | 6141 |

Table 9: Numeric Results for Figure 6(b).
### Table 10: Numeric Results for Figure 7(a).

| Model        | $\mu=0$ | $\mu=0.5$ | $\mu=1.0$ | $\mu=2.0$ | $\mu=5.0$ | $\mu=10$ | DB  |
|--------------|---------|-----------|-----------|-----------|-----------|---------|-----|
| Wait-6-Stride-2 | 24.50   | 24.64     | 24.79     | 24.61     | 24.35     | 24.13   | 24.24 |
| Wait-Inf     | 27.28   | 27.32     | 27.40     | 27.31     | 27.28     | 27.15   | 26.89 |

### Table 11: Numeric Results for Figure 7(b).

| Model                        | K=2 BLEU | K=2 AL | K=6 BLEU | K=6 AL | K=10 BLEU | K=10 AL | K=inf BLEU | K=inf AL |
|------------------------------|----------|--------|----------|--------|-----------|---------|------------|----------|
| RealTranS                    | 18.40    | 1064   | 24.35    | 2070   | 25.79     | 2963    | 26.93      | 6141     |
| 4 blocks w/o shrink          | 13.03    | 307    | 19.25    | 1207   | 21.55     | 2051    | 25.38      | 6141     |
| 2 blocks with shrink         | 12.53    | 1247   | 19.48    | 2286   | 21.97     | 3175    | 25.04      | 6141     |

### Table 12: Numeric Results for Figure 8.

| Model                     | K=2 BLEU | K=2 AL | K=6 BLEU | K=6 AL | K=10 BLEU | K=10 AL | K=inf BLEU | K=inf AL |
|---------------------------|----------|--------|----------|--------|-----------|---------|------------|----------|
| Full Model                | 18.45    | 1023   | 24.79    | 2043   | 25.82     | 2920    | 27.40      | 6141     |
| -CTC PT                   | 18.40    | 1064   | 24.35    | 2070   | 25.79     | 2963    | 26.93      | 6141     |
| -BP                       | 20.20    | 1255   | 24.61    | 2393   | 25.53     | 3325    | 26.11      | 6141     |
| -Shrink                   | 20.10    | 1368   | 24.38    | 2426   | 25.22     | 3357    | 26.49      | 6141     |
| -GD                       | 19.66    | 1360   | 24.11    | 2390   | 25.01     | 3316    | 25.56      | 6141     |
| -CTC                      | 12.48    | 342    | 18.42    | 1230   | 21.09     | 2055    | 24.73      | 6141     |