End-to-End Simultaneous Speech Translation with Pretraining and Distillation: Huawei Noah's System for AutoSimTranS 2022

Xingshan Zeng, Pengfei Li, Liangyou Li, Qun Liu
Huawei Noah's Ark Lab
{zeng.xingshan,lipengfei111,liliangyou,qun.liu}@huawei.com

Abstract

This paper describes the system submitted to AutoSimTrans 2022 from Huawei Noah's Ark Lab, which won the first place in the audio input track of the Chinese-English translation task. Our system is based on RealTranS, an end-to-end simultaneous speech translation model. We enhance the model with pretraining, by initializing the acoustic encoder with ASR encoder, and the semantic encoder and decoder with NMT encoder and decoder, respectively. To relieve the data scarcity, we further construct pseudo training corpus as a kind of knowledge distillation with ASR data and the pretrained NMT model. Meanwhile, we also apply several techniques to improve the robustness and domain generalizability, including punctuation removal, token-level knowledge distillation and multi-domain finetuning. Experiments show that our system significantly outperforms the baselines at all latency and also verify the effectiveness of our proposed methods.

1 Introduction

Simultaneous Speech Translation (ST) task (Fügen et al., 2007; Oda et al., 2014) aims to translate speech into the corresponding text in another language while reading the source speech. Prior works mainly focus on the cascaded solution, i.e., first recognize the speech with a streaming ASR model and then translate into the target language with simultaneous NMT (Ma et al., 2019) model. Such cascaded systems can leverage off-the-shelf ASR and NMT systems, which have large-scale data for training.

Recently, end-to-end simultaneous ST models are also proposed (Ren et al., 2020; Zeng et al., 2021) and have shown promising improvements towards cascaded models when experimented on the same amount of data, especially in low latency requirement. End-to-end models are believed to have the advantages of lower latency, smaller model size and less error propagation (Weiss et al., 2017), but suffer from data scarcity. A well-trained end-to-end model typically needs a large amount of training data. To alleviate the data scarcity problem, pretraining (Xu et al., 2021; Li et al., 2021) and data augmentation (Bahar et al., 2019; Jia et al., 2019) are two main techniques. We examine the effectiveness of the two techniques for improving end-to-end models in this work.

Specifically, our end-to-end ST model follows RealTranS (Zeng et al., 2021), an encoder-decoder model and the encoder is decoupled into acoustic encoder and semantic encoder. The acoustic encoder is used to extract acoustic features which has a similar function as the ASR encoder. Therefore we initialize it with a pretrained ASR encoder. The semantic encoder is required to learn semantic knowledge, which benefits the translation task, so we initialize it with a pretrained NMT encoder. The decoder is also initialized with a pretrained NMT decoder to produce target text decoding. For data augmentation, we construct pseudo ST corpus based on ASR data and the pretrained NMT model. The ground-truth transcription is translated into target language texts, and so speech-transcription-translation triplets for ST training are built. This is also known as sequence-level knowledge distillation (Kim and Rush, 2016). Generally, the NMT data can also be augmented with a TTS model to generate pseudo speech. However, the data quality highly depends on the TTS performance and it is hard for TTS to produce voices similar to those in real scenarios. Thus we do not utilize this method and leave it to the future work. Another popular technique for audio data augmentation is SpecAugment (Park et al., 2019; Bahar et al., 2019), which randomly masks a block of consecutive time steps and/or mel frequency channels of the input speech features during training. It is a simple and low-implementation cost method and has been shown effective in avoiding overfitting and improving ro-

Proceedings of the Third Workshop on Automatic Simultaneous Translation, pages 25 - 33

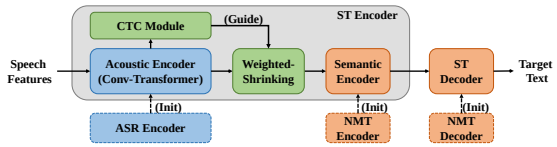


Figure 1: Our RealTranS model with pretraining.

bustness. We apply it to all audio-related model training.

The training procedure for our ST model mainly contains three steps: ASR and NMT pretraining, large-scale training on the constructed pseudo data, and finetuning on the in-domain data. During training, we remove all the punctuation in source text in audio-related training (i.e., excluding NMT pretraining) to relieve the learning burden and improve recognition quality. To enhance the final performance after finetuning, we also utilize token-level knowledge distillation from the full-sentence NMT model and multi-domain finetuning trick.

Our model is used to participate in the audio input track of the AutoSimTrans 2022 Chinese-English translation task. In this track, an in-domain ST data called BSTC (Zhang et al., 2021) (contains about 70 hours of audios) is provided, which is very limited. Therefore, assisted with extra ASR and NMT data, we use the aforementioned techniques to achieve remarkable improvement at all latency requirement, which results in winning the first place of the track. We also conduct more experiments to examine the effectiveness of our used techniques. The experiments show that all of our used methods contribute to the improvement of the final model.

2 Model Description

We build our model based on RealTranS (Zeng et al., 2021), an end-to-end simultaneous speech translation model with its encoder decoupled into acoustic encoder and semantic encoder (see Figure 1). With a CTC module guiding the acoustic encoder to produce acoustic-level features, the decoupling relieves the burden of the ST encoder and makes the two separate modules focus on different knowledge, which benefits the model training.

RealTranS leverages the unidirectional Conv-Transformer (Huang et al., 2020) as the acoustic encoder for gradual downsampling, and weighted-shrinking for bridging the modality gap between speech and text. With weighted-shrinking, long speech features are shrunk to similar lengths as their corresponding transcription, which makes the input of the semantic encoder more similar to the input of NMT encoder. In this way, the difficulty of knowledge transferring when we initialize the semantic encoder with NMT encoder becomes smaller. Apart from the semantic encoder, we also initialize the acoustic encoder with pretrained ASR encoder, and the decoder with pretrained NMT decoder, which has been shown very useful in boosting the performance (Xu et al., 2021).

For simultaneous policy, we use the wait-k-stride-n policy (Zeng et al., 2021), which has shown promising improvement over the conventional wait-k policy (Ma et al., 2019).

3 Training Procedure

Our model training consists of three steps: ASR and NMT pretraining, large-scale training on the constructed pseudo data, and finetuning on the in-domain data. Each step may contain different techniques to enhance model performance and we will describe them in-detailed as follows.

3.1 Pretraining

We first describe how we pretrain our ASR and NMT models.

ASR Pretraining. Our ASR model follows the architecture of Conv-Transformer Transducer proposed by Huang et al. (2020). A Transducer model contains an audio encoder, a prediction net and a joint net, where the audio encoder is used for initializing the acoustic encoder of our ST model and the rest discarded. For each frame in input speech features, the model first predicts either a token label from the vocabulary or a special blank symbol. When a label is predicted, the model continues to predict the next output; when the model predicts a blank symbol, it proceeds to the next frame indicating no more labels can be predicted with current frames. Therefore, for each input speech $\boldsymbol{x}$, the model will give $T_x + T_z$ predictions, where T_x (the length of $\boldsymbol{x}$) is the number of blank symbols and T_z is the number of token labels representing the output transcription $\boldsymbol{z}$. A Transducer model computes the following marginalized distribution and maximizes it during training:

$$p(\boldsymbol{z}|\boldsymbol{x}) = \sum_{\hat{\boldsymbol{z}} \in \mathcal{A}(\boldsymbol{x},\boldsymbol{z})} \prod_{i=1}^{T_x+T_z} p(\hat{z}_i|x_1, ..., x_{t_i}, z_0, ..., z_{u_i-1}) \tag{1}$$

where $\mathcal{A}(\boldsymbol{x}, \boldsymbol{z})$ is the set containing all valid alignment paths such that removing the blank symbols

in $\hat{\boldsymbol{z}}$ yields $\boldsymbol{z}$. The summation of probabilities of all alignment paths is computed efficiently with forward-backward algorithm.

As there is no ASR data provided, we collect large-scale ASR datasets from both publicly available websites and our internal system (the statistics of the datasets are in Table 1) for training. During training, we also add additive Gaussian noise and apply speed perturbation (Ko et al., 2015) and SpecAugment (Park et al., 2019) for data augmentation and model robustness.

Finally, our pretrained ASR model gets the performance of 11.35% WER (Word Error Rate) in BSTC development set.

NMT Pretraining. We pretrain our NMT model with CeMAT (Li et al., 2022), a sequence-to-sequence pretraining model but with a bidirectional decoder, which has been shown to be effective in NMT tasks. CeMAT can be pretrained on large-scale bilingual and monolingual corpus. As no additional text data are available, we only use the dynamic dual-masking algorithm to improve performance. Given an input source sentence $\boldsymbol{z}$, we first sample a masking ratio μ from a uniform distribution between $[0.1, 0.2]$, then randomly mask a subset of source words according to μ. For the corresponding target sentence $\boldsymbol{y}$, we also use a uniform distribution between $[0.2, 0.5]$ to sample a masking ratio υ. Following CeMAT, we set $\upsilon \geq \mu$ to force the bidirectional decoder to obtain more information from the encoder. For monolingual, we create pseudo bilignual text by copying the sentence, then sample $\upsilon = \mu$ from a uniform distribution between $[0.3, 0.4]$ and mask the same subset on both sides. After dual-masking, we get the new sentence pair $(\hat{\boldsymbol{z}}, \hat{\boldsymbol{y}})$, which will be used for jointly training the encoder and decoder by predicting masked tokens on both sides. The final training objective is formulated as follows:

$$\begin{aligned}\mathcal{L} = -\sum_{(\hat{\boldsymbol{z}},\hat{\boldsymbol{y}})} \lambda \sum_{y_j \in \boldsymbol{y}^{mask}} \log P(y_j|\hat{\boldsymbol{z}}, \hat{\boldsymbol{y}}) \\ +(1-\lambda) \sum_{z_i \in \boldsymbol{z}^{mask}} \log P(z_i|\hat{\boldsymbol{z}})\end{aligned} \tag{2}$$

where $\boldsymbol{y}^{mask}$ are the set of masked target words, $\boldsymbol{z}^{mask}$ are the set of masked source words, and λ is a hyper-parameter to balance the influence of both sides. Following CeMAT, we set $\lambda = 0.7$.

Our NMT pretraining procedure can be summarized as three sub-steps. We first train a basic NMT model using the provided general-domain bilingual data (see Table 1), and generate pseudo target sentences based on the source text from the ASR data used in ASR pretraining. To improve the quality of the pseudo corpus, we use HintedBT (Ramnath et al., 2021) to score each generated sentences. Next, we combine the bilingual data, the pseudo corpus and the monolingual text (from the used ASR data) to pretrain CeMAT. Finally, we finetune it on the bilingual and pseudo corpus including the in-domain data (i.e. text part in BSTC dataset) to produce our final NMT model.

The encoder and decoder of the NMT model is used to initialize the semantic encoder and decoder of our ST model, respectively. It is also used to generate pseudo ST data in next subsection.

Our NMT model achieves BLEU score of 21.82 in BSTC development set, and also won the second place in the streaming transcription input track of the Chinese-English translation task.

3.2 Training on Pseudo Data (Distillation)

As the provided ST data is limited (about 70 hours annotated data), it is difficult to directly train an end-to-end model only with the provided data. We decide to construct pseudo data from our used ASR data – we translate the Chinese transcription into English translation with our pretrained NMT model so that we get a large-scale pseudo ST corpus with audio-transcription-translation triplets. In this way, we can leverage large-scale unannotated audios and distill knowledge from the NMT model. We remove all the punctuation in transcription (as the ASR data comes from different domains, some of them contain punctuation but some not) to make it consistent during training.

We first train our model (initialized with the pretrained modules described in Section 3.1) on the pseudo data with multi-path wait-k training (Elbayad et al., 2020) to cover all possible k values. Specifically, k value will be uniformly sampled from $K = [1, ..., |K|]$ for each training sample during training while we keep the n value in wait-k-stride-n policy at 2. In this way, the model can learn knowledge for different latency requirements. The training objectives follows RealTranS and contain the CTC loss ($\mathcal{L}_{CTC}$) (Graves et al., 2006) with a blank penalty ($\mathcal{L}_{BP}$) (Zeng et al., 2021). We omit their equations here and refer the readers to Zeng et al. (2021) for details. The translation loss are defined as follows:

$$\mathcal{L}_{ST} = -\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D},k\sim\mathcal{U}(K)} \prod_{t=1}^{T_y} p(y_t|y_{<t}, x'_{\leq g_{k,n}(t)}) \quad (3)$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input speech features and the output token sequence, and $\mathcal{D}$ is the training corpus. $y_{<t}$ denotes the target tokens before time step t and $x'_{\leq g_{k,n}(t)}$ represents the first $g_{k,n}(t)$ source features after weighted-shrinking (generally, one shrunk feature may represent one source token as they are shrunk based on CTC output probability) with $g_{k,n}(t) = n\lfloor(t-1)/n\rfloor + k$. Finally, the total training objective is:

$$\mathcal{L}_{PT} = \mathcal{L}_{ST} + \alpha\mathcal{L}_{CTC} + \beta\mathcal{L}_{BP} \quad (4)$$

where α and β are the hyper-parameters to balance the losses, which are set to $\alpha = 1.0$ and $\beta = 0.5$. During training, we also apply SpecAugment.

3.3 Finetuning on In-Domain Data

After the large-scale training on pseudo data, we use the in-domain data (i.e., the provided 70 hours BSTC data) for finetuning. To be consistent with the training in the previous step, we also remove all punctuation in the source texts for CTC loss. During finetuning, there are mainly two aspects that are different from the large-scale training in the previous step. First, we fix the k value rather than use the multi-path wait-k training and train several models with different k as the in-domain data is very small. Second, we add a token-level knowledge distillation (KD) loss guided by the full-sentence NMT model pretrained in Section 3.1. Note that the input for the NMT model are the source texts with punctuation preserved. In this way, the final ST model can also learn from text translation. The KD loss is defined as follows:

$$\mathcal{L}_{KD} = -\sum_{(\boldsymbol{x},\boldsymbol{z},\boldsymbol{y})\in\mathcal{D}_{ST}} \sum_{t=1}^{T_y}\sum_{k=1}^{|V|} q(y_t = v_k|y_{<t},\boldsymbol{z}) \times \log p(y_t = v_k|y_{<t}, x'_{\leq g_{k,n}(t)}) \quad (5)$$

where $\boldsymbol{x}$, $\boldsymbol{z}$ and $\boldsymbol{y}$ are the input speech features, the input source texts and the output token sequence, and $\mathcal{D}_{ST}$ is the in-domain training corpus. Therefore, the total finetuning objective is:

$$\mathcal{L}_{FT} = (1-\gamma)\mathcal{L}_{ST} + \gamma\mathcal{L}_{KD} + \alpha\mathcal{L}_{CTC} + \beta\mathcal{L}_{BP} \quad (6)$$

where γ controls the tradeoff between the ST and KD losses and is set to 0.2. We set $\alpha = 1.0$ and $\beta = 0.5$.

Dataset	SRC Speech	SRC Text	TGT Text	#Hours	#Sents
CWMT21	×	✓	✓	–	9M
Internal	✓	✓	Pseudo	10K	11M
WenetSpeech	✓	✓	Pseudo	10K	14M
BSTC	✓	✓	✓	70	38K

Table 1: The statistics of the used datasets.

Note that in our experiments, we utilize multi-domain finetuning rather than finetuning only with the in-domain data, i.e., we also randomly sample similar number of training samples from the constructed pseudo data for finetuning. This improves domain generalizabiliy of our model. More analysis can be found in Section 4.3.

4 Experiments

4.1 Experimental Setup

Datasets. We introduce the details of the datasets we use here. Table 1 displays the statistics of them. CWMT21[1] is the NMT data in general domain provided by the organizer. We mainly use it to pretrain our NMT model. Internal and WenetSpeech are large-scale ASR datasets, both of which contain about 10K hours of audios. WenetSpeech is mainly collected from YouTube and Podcast and is publicly available[2], while Internal comes from our internal system, containing conversations or readings from multiple domains. The transcription in them are translated into target texts with our pretrained NMT model, which results in large-scale pseudo ST data. Finally, BSTC is the in-domain ST data provided by the organizer. It is used to finetune our NMT model (with source texts and target texts) and the final ST model.

The punctuation in source texts is processed with different ways according to different training procedure (see Section 3 for details), while that in target texts is always preserved. We also filter the data based on the lengths of source and target texts. We follow Zhang and Feng (2021) and use char-level tokenization on the Chinese sentences, while we apply sentencepiece[3] (Kudo and Richardson, 2018) to generate subword vocabulary for English.

System Setting. We use 128-dimensional log-mel filterbank as acoustic features, calculated with 20 ms window and 10 ms stride and normalized by

[1] http://mteval.cipsc.org.cn:81/agreement/AutoSimTrans

[2] https://wenet.org.cn/WenetSpeech/

[3] https://github.com/google/sentencepiece

Global CMVN (cepstral mean and variance normalization). Our acoustic encoder contains two blocks of Conv-Transformer (Huang et al., 2020). In the first block, we have two 2-D convolution layers with stride 2 and four layers of transformer encoder; while in the second block, we have three 1-D convolution layers with one of them is stride 2 (others 1) and 16 layers of transformer encoder. This results in total $8\times$ downsampling and introduces a 100ms look-ahead window. In the first block, transformer layers are with 384 dimension and 6 attention heads, while in the second block they are with 512 dimension and 8 attention heads. For the semantic encoder and decoder, as they are initialized with the NMT model, they follow the deep encoder, shallow decoder architecture to improve inference efficiency, with 12 encoder layers, 3 decoder layers, 12 attention heads and 768 hidden size.

Our model is trained with 24 NVIDIA Tesla V100 GPUs, each with a max-tokens of 2048 (i.e., maximum of 2048 text tokens in one batch). We use Adam optimizer (Kingma and Ba, 2015) during model training with $2e^{-3}$ learning rate and 10000 warm-up steps, followed by the inverse square root scheduler. Dropout rate is set to 0.1. For SpecAugment, we set the parameter for time masking T to 40 and that for frequency masking F to 4. The number of time and frequency masks applied m_T and m_F are 2 and 1, respectively.

Evaluation Metrics. For evaluation, we use case-sensitive detokenized SacreBLEU[4] for translation quality evaluation. For latency, we adapt Average Lagging (AL) (Ma et al., 2019) to ST settings, following previous studies (Ma et al., 2020; Zeng et al., 2021). In the submission system, the latency is evaluated with Consecutive Wait (CW) (Gu et al., 2017).

AL in ST evaluates the degree of that the user is out of sync with the speaker, in terms of source speech time duration (Zeng et al., 2021), which is defined as follows:

$$AL(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\tau(|\boldsymbol{x}|)} \sum_{i=1}^{\tau(|\boldsymbol{x}|)} [d(y_i) - \frac{|\boldsymbol{x}|}{|\boldsymbol{y}^*|} T_s(i-1)] \quad (7)$$

where $\tau(|\boldsymbol{x}|)$ denotes the target token index when the model has read the entire source speech. $|\boldsymbol{y}^*|$ is the length of the reference translation, and T_s represents that the speech features are extracted every T_s ms, which will be 80ms in our model. As our acoustic encoder introduces a 100ms look-ahead window, we add 100 to the final AL scores.

CW is the number of source tokens waited between two target tokens, which can be calculated with the following equation (Ma et al., 2019):

$$CW(\boldsymbol{x}, \boldsymbol{y}) = \frac{|\boldsymbol{x}|}{\sum_{i=1}^{|\boldsymbol{y}|} \mathbb{1}_{CW_g(t)>0}} \quad (8)$$

where $\boldsymbol{x}$ here is the corresponding transcription of source speech, and $CW_g(t)$ denotes the waiting source token numbers between time step $t-1$ and t. This means that when evaluated with CW, our model also needs to output transcription. We decide to output the results of CTC greedy paths. In this way, CW can be easily calculated as our wait-k-stride-n policy is applied on the shrunk speech features which are also based on the CTC module.

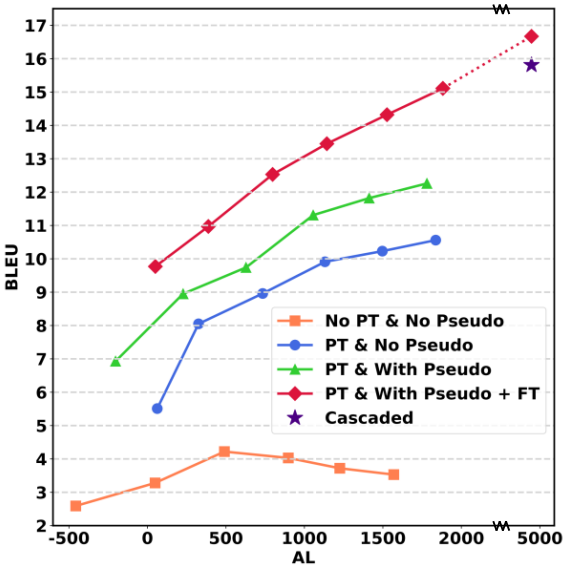


Figure 2: Comparison results of our model variants.

4.2 Main Results

Figure 2 displays the comparison results among different variants of our model. We compare the following settings:
(a) *No PT & No Pseudo*: We do not use any pretrained modules and directly train the model on the provided in-domain data.
(b) *PT & No Pseudo*: We initialize the model with the pretrained modules and directly train the model on the provided in-domain data.
(c) *PT & With Pseudo*: We initialize the model with the pretrained modules and train the model on the large-scale constructed pseudo data.
(d) *PT & With Pseudo + FT*: We further finetune the model on the in-domain data with multi-domain

[4] https://github.com/mjpost/sacreBLEU

Model	CTC Loss (↓)		BLEU (↑)	
	With Punct	RM Punct	With Punct	RM Punct
No PT & No Pseudo	4.88	4.60	4.12	4.03
FT based on PT + Pseudo	2.10	1.73	11.81	12.41

Table 2: CTC Loss and BLEU results of models trained on in-domain BSTC data, with punctuation preserved or removed. We set the same k ensuring similar latency.

finetuning based on model c.

All of the model variants use wait-k-stride-n simultaneous policy with n=2 and k=2, 4, 6, 8, 10, 12, respectively.

We also compare with the performance of our cascaded model, which first passes the audio input into our pretrained ASR model and then translates with our pretrained NMT model. Since the NMT model is an offline full-sentence translation model, the latency is much higher than the other variants (about 5000ms AL). Therefore, we also compute the offline translation result of our model d for fair comparison.

As can be seen, without any pretraining and extra data, the model performs poorly (model a). With pretraining (model b) and large-scale pseudo data (model c), the model performance increases significantly, which validates the effectiveness of the two training tricks Further finetuning with in-domain data (model d) also introduces reasonable improvement, which shows the importance of domain adaptation. Compared to the cascaded result, our model achieves almost 1 BLEU better than it, indicating the superiority of our RealTranS end-to-end model.

4.3 Further Analysis

Effects of Punctuation Removal. We mainly examine the effects of punctuation removal (only for transcription) during training on the in-domain data. We experiment with two settings. The first one is to directly train on the in-domain data without any pretraining or pseudo data, and the second one is finetuning based on the model with pretraining and pseudo data. We display the results of them with and without punctuation in Table 2.

Both models achieve lower CTC loss values with punctuation removal. It validates that the model can learn better on the acoustic information when punctuation is removed. However, no significant difference is observed in BLEU for the first model while the BLEU is degraded when finetuning the pretrained model with punctuation preserved. It is mainly because the model is first pretrained on the data without punctuation and can be trained more smoothly without punctuation when finetuning.

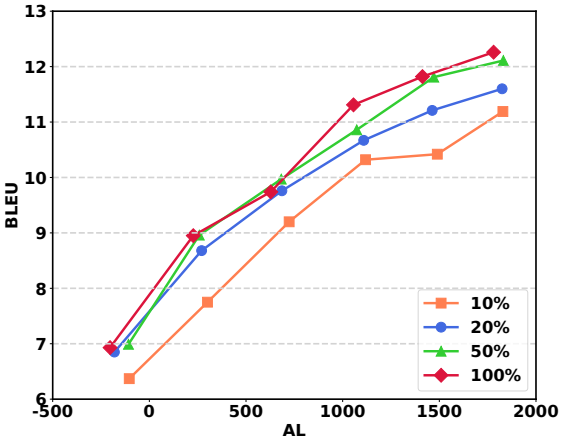


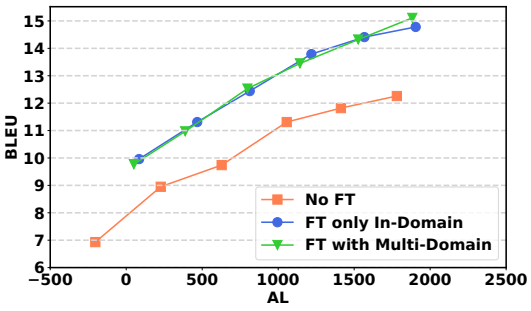
Figure 3: Results of our model when using different amount of pseudo data.

λ value	0.0	0.2	0.4	0.6	0.8	1.0
BLEU	13.07	**13.79**	13.70	13.39	12.53	11.06

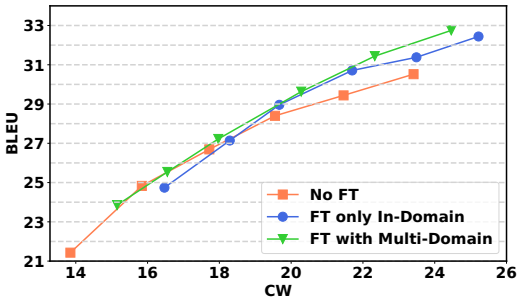
Table 3: BLEU scores of models trained with different λ values in Eq. 6. We set the same k ensuring similar latency.

Effects of Pseudo Data Amount. We also want to examine the effects of pseudo data amount during training. We sample 10%, 20% and 50% of our constructed pseudo data and then use them to train our model, respectively. Figure 3 shows the results (before finetuning), together with our model with full data (100%). Comparing models trained with 10% and 20%, it introduces sufficient improvement when doubling the pseudo data. However, when the pseudo data continues to increase, the performance gain becomes smaller, especially for the results in low latency. It is probably because that our NMT model used for generating pseudo translations is trained with limited NMT data (around 9M sentences), much smaller than the used ASR data (around 25M, according to Table 1). The amount of knowledge it carries is not enough to provide such larger amount of efficient pseudo data. Therefore, we might need large amount of data for both recognition and translation to train a more powerful end-to-end ST model.

Effects of Token-level Knowledge Distillation. Our token-level knowledge distillation (KD) from full-sentence NMT model can guide the learning of ST model to forecast target text when the source speech is incomplete during finetuning. We examine the effects of the balance value λ in Eq. 6. The



(a) Results in Development Set



(b) Results in Test Set

Figure 4: Results with different finetuning methods.

results are displayed in Table 3. $\lambda = 0.0$ indicates that no KD is used, and $\lambda = 1.0$ means the model is trained only based on KD but no cross-entropy loss. It can be found that too much guidance from the KD might hurt the performance. We attribute this to two reasons. First, the NMT model is mainly trained with CWMT21 data, which is limited and in a different domain. Second, our model has already been trained with the constructed pseudo data, which can be viewed as another kind of KD (sequence-level KD). Therefore, we choose to select smaller λ (i.e., 0.2) in our experiments.

Effects of Multi-Domain Finetuning. Figure 4 shows the results of our model without finetuning (No FT), only finetuned with in-domain data (FT only In-Domain) and finetuned with multi-domain corpus (FT with Multi-Domain) in the development set and test set, respectively[5]. We can find that though finetuning only with the in-domain data improves a lot in the development set (in average nearly 2 BLEU gain at each latency requirement), the improvement in the test set is limited and performance even hurts at one latency setting (No PT v.s. FT only In-Domain). We attribute this to the fact that the test set may be not exactly in the same domain as the training and development data, and the naive finetuning degrades the ability of domain generalizability. Therefore, we decide to use multi-domain finetuning rather than finetuning only with the in-domain data. As can be seen in Figure 4, this brings no improvement in the development set (FT with Multi-Domain v.s. FT only In-Domain), but improves in the test set.

[5]Note that the test results are validation experiments afterwards and not our submission results.

5 Conclusion

In this work, we describe the details of our submitted system to AutoSimTranS 2022, which won the first place in Chinese-English audio input track. Our model is based on the end-to-end simultaneous speech translation model RealTranS and follows three-step training procedure, including ASR and NMT pretraining, large-scale training on the pseudo data and finetuning on the in-domain data. Our experiments proves the superiority of our model and training procedure and also examines the effectiveness of different techniques like punctuation removal and multi-domain finetuning.

References

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

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2020. Efficient wait-k models for simultaneous machine translation. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 1461–1465. ISCA.

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

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)*, 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 5001–5005. ISCA.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J. Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. 2019. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 7180–7184. IEEE.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3586–3589. ISCA.

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.

Pengfei Li, Liangyou Li, Meng Zhang, Minghao Wu, and Qun Liu. 2022. Universal conditional masked language pre-training for neural machine translation. *CoRR*, abs/2203.09210.

Xian Li, Changhan Wang, Yun Tang, Chau Tran, Yuqing Tang, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2021. Multilingual speech translation from efficient finetuning of pretrained models. 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 827–838, Online. 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, Juan Pino, and Philipp Koehn. 2020. 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.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 551–556, Baltimore, Maryland. Association for Computational Linguistics.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2613–2617. ISCA.

Sahana Ramnath, Melvin Johnson, Abhirut Gupta, and Aravindan Raghuveer. 2021. Hintedbt: Augmenting back-translation with quality and transliteration hints. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1717–1733. Association for Computational Linguistics.

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.

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.

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.

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.

Ruiqing Zhang, Xiyang Wang, Chuanqiang Zhang, Zhongjun He, Hua Wu, Zhi Li, Haifeng Wang, Ying Chen, and Qinfei Li. 2021. BSTC: A large-scale Chinese-English speech translation dataset. In *Proceedings of the Second Workshop on Automatic Simultaneous Translation*, pages 28–35, Online. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2021. ICT's system for AutoSimTrans 2021: Robust char-level simultaneous translation. In *Proceedings of the Second Workshop on Automatic Simultaneous Translation*, pages 1–11, Online. Association for Computational Linguistics.