The YiTrans End-to-End Speech Translation System for IWSLT 2022 Offline Shared Task

Ziqiang Zhang[1]*, **Junyi Ao**[2,*]
[1]School of Information Science and Technology,
University of Science and Technology of China
[2]School of Data Science, The Chinese University of Hong Kong (Shenzhen)

Abstract

This paper describes the submission of our end-to-end YiTrans speech translation system for the IWSLT 2022 offline task, which translates from English audio to German, Chinese, and Japanese. The YiTrans system is built on large-scale pre-trained encoder-decoder models. More specifically, we first design a multi-stage pre-training strategy to build a multi-modality model with a large amount of labeled and unlabeled data. We then fine-tune the corresponding components of the model for the downstream speech translation tasks. Moreover, we make various efforts to improve performance, such as data filtering, data augmentation, speech segmentation, model ensemble, and so on. Experimental results show that our YiTrans system obtains a significant improvement than the strong baseline on three translation directions, and it achieves +5.2 BLEU improvements over last year's optimal end-to-end system on tst2021 English-German.

1 Introduction

In this paper, we describe our end-to-end speech translation system YiTrans which participates in the offline tracks of the IWSLT 2022 evaluation campaign. We evaluate our systems from English to German, Chinese and Japanese. We aim at exploring the pre-training methods for end-to-end systems, and bridging the quality gap with the cascaded approaches.

As self-supervised learning has been shown effective in speech-to-text tasks (Baevski et al., 2020; Hsu et al., 2021; Ao et al., 2021; Bapna et al., 2021), our teams are interested in building a multi-modality pre-trained model with self-supervised approaches by leveraging large amounts of speech and text data. Inspired by SpeechT5 (Ao et al., 2021), we design a multi-stage unified-modal training strategy for pre-training both the encoder and decoder. Our final end-to-end ST systems are built by fine-tuning the pre-trained models.

This paper also tries to improve the system performance by exploring various techniques for the related tasks. (1) To boost the performance with advanced speech segmentation (Anastasopoulos et al., 2021), we apply the pyannote toolkit (Bredin et al., 2020) and the merge algorithm from Inaguma et al. (2021) to segment the audio. Particularly, to overcome the long sentence problem in the dataset, we design a new segment algorithm. (2) Dataset is the key point for a ST system to perform well. Hence, we conduct refined data filtering and large-scale data augmentation (Jia et al., 2019). (3) We also employ progressive learning, back translation and multi-stage fine-tuning (Yang et al., 2021; Sennrich et al., 2015; Wang et al., 2020b) when fine-tuning our models. (4) Motivated by Tang et al. (2021a), we utilize joint ST and MT fine-tuning for our end-to-end ST models. (5) As comparison, we also build the cascaded systems for all three language pairs by fine-tuning ASR and MT models from pre-trained models.

The rest of this paper is organized as follows. In Section 2, we describe the data preparation, including the data pre-processing, data augmentation, and speech segmentation. Section 3 illustrates the unified-modal pre-training methods, and our systems for all three tasks. We share the experimental setting, results, and analyses in Section 4. Section 5 concludes the submission. We also present the official test results (Anastasopoulos et al., 2022) of our submitted system in Appendix A.

2 Data Preparation

2.1 Datasets

Our system is built under constraint conditions. The training data can be divided into five categories: unlabeled audio, monolingual text, ASR, MT, and

*Equal contribution.

Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022), pages 158 - 168

ST corpora1.

Datasets	# Utterances	# Hours
Unlabeled Data		
VoxPopuli	1224.9k	28708
Labeled ASR Data		
MuST-C v1&v2	341.6k	616.9
ST-TED	171.1k	272.8
LibriSpeech	281.2k	961.1
CoVoST	288.4k	426.1
CommonVoice	1224.9k	1668.1
TEDLIUM v2&v3	361.2k	660.6
Europarl	34.3k	81.4
VoxPopuli ASR	177.0k	501.3
Labeled ST Data		
en-de		
MuST-C v2	249.8k	435.9
ST-TED	171.1k	272.8
CoVoST	288.4k	426.1
Europarl	32.6k	77.2
en-ja		
MuST-C v2	328.4k	534.5
CoVoST	288.4k	426.1
en-zh		
MuST-C v2	358.5k	586.8
CoVoST	288.4k	426.1

Table 1: English audio data statistics

Unlabeled Audio We utilize large-scale unlabeled and labeled audio for pre-training. As shown in Table 1, we pre-train our models by using around 28k hours of unlabeled audio data from VoxPopuli (Wang et al., 2021), and around 5.1k hours of labeled ASR data, which will be introduced later.

Monolingual Text Monolingual text is used either for pre-training or back-translation. We collect data for English as well as three target languages from WMT21 news translation task[1], including News Commentary[2], Europarl v10[3], News crawl[4], and Common Crawl[5]. As Common Crawl contains much noisier data, it is only used for **ja** and **zh** to expand the collected data size to 500M. The statistics are listed in Table 2.

[1]https://www.statmt.org/wmt21/translation-task.html
[2]http://data.statmt.org/news-commentary
[3]http://www.statmt.org/europarl/v10
[4]http://data.statmt.org/news-crawl
[5]http://data.statmt.org/ngrams

	en	de	ja	zh
Collected	341M	389M	500M	500M
Processed & filtered	50M	50M	50M	50M

Table 2: Monolingual text data statistics

ASR Corpus For training and evaluation of our ASR models, we use MuST-C v1 (Di Gangi et al., 2019), MuST-C v2 (Cattoni et al., 2021), ST-TED (Niehues et al., 2018), LibriSpeech (Panayotov et al., 2015), CoVoST 2 (Wang et al., 2020a), TED-LIUM v2 (Rousseau et al., 2012), TED-LIUM v3 (Hernandez et al., 2018), Europarl (Koehn, 2005), VoxPopuli ASR data, and Mozilla Common Voice (Ardila et al., 2019), which results in around 5188.3hr labled ASR data as shown in Table 1. For MuSTC-C and Europarl, we collected the data from all language pairs and removed the overlap audios according to the audio id.

Datasets	en-de	en-ja	en-zh
In-domain			
MuST-C v2	249.8k	328.4k	358.5k
TED	209.5k	223.1k	231.3k
Out-of-domain			
CoVoST	288.4k	288.4k	288.4k
Europarl	32.6k	-	-
OpenSubtitles2018	18.7M	1.9M	10.0M
WMT21	93.3M	16.6M	61.0M
Sum (processed)	82.0M	13.8M	51.5M
Sum (filtered)	16.1M	3.6M	7.6M

Table 3: MT data statistics

MT Corpus Machine translation (MT) corpora are used to translate the English transcription. For training and evaluation of our MT models, we use MuST-C v2 and TED corpus (Cettolo et al., 2012) as in-domain data. We also use CoVoST 2, Europarl, OpenSubtitles2018 (Lison and Tiedemann, 2016) as well as all available paired data provided by WMT21 as out-of-domain data. The statistics are listed in Table 3.

ST Corpus The ST corpus we used includes the MuST-C v2, ST-TED, CoVoST 2 and Europarl, as listed in Table 1. MuST-C v2 and ST-TED are treated as in-domain data. The ST corpus can be greatly expanded by large-scale data augmentation,

which will be introduced in the following Section.

2.2 Text Processing & Filtering

For monolingual and out-of-domain MT data, we first process the text through the following steps:

(1) We clean up the data by removing sentences that have non-printing characters, http tags or words with length longer than 50 characters (words are separated by space, for **ja** and **zh** the threshold is 150). The processed text data is then deduplicated.

(2) We use fast-text [6] (Joulin et al., 2016) to filter out the sentences with invalid languages.

(3) For paired data, we use fast_align[7] (Dyer et al., 2013) to calculate the alignment quality, which is evaluated by the percentage of aligned words. We remove 20% of data with the lowest alignment quality.

(4) We then use XenC[8] (Rousseau, 2013) to perform domain filtering. It computes the distinction of two n-gram language models, which are in-domain and out-of-domain language models. The amount of selected data is 50M for monolingual text, and for paired text it depends on the XenC scores. The results are listed in Table 2 and 3.

2.3 Post processing

We only do post-processing for **en-ja** systems as an optional choice. It is because we noticed that for **en-ja** there is few punctuations in the target side of training data. To obtain translation results with rich punctuation, which are more natural in the real world, we train a punctuation model to post-process the translated results. The model is initialized from mBART50 (Tang et al., 2020) and trained to predict sentences with proper punctuation. The training data is collected from out-of-domain **en-ja** MT data. We select the sentences with rich punctuation in Japanese side.

2.4 Data Augmentation

The quality of end-to-end ST is often limited by a paucity of training data, since it is difficult to collect large parallel corpora of speech and translated transcript pairs In this paper, we attempt to build a large amount of synthetic data for ST and MT, separately. We will introduce the data augmentation method in Section 3 in detail.

[6]https://github.com/facebookresearch/fastText

[7]https://github.com/clab/fast$_a lign$

[8]https://github.com/antho-rousseau/XenC

2.5 Speech Segmentation

Algorithm 1 Segment audios based on pyannote toolkit

function SEGMENTAUDIO($x, P_{on}, P_{off}, T_{dur}$)
  $L \leftarrow VAD(x, P_{on}, P_{off})$ ▷ $\{a_1, ..., a_n\}$
  $L_{new} \leftarrow \{\}$
  for $a_i \in L$ **do**
    if $a_i.length > T_{dur}$ **then**
      if $P_{on} < 0.95$ or $P_{off} < 0.95$ **then**
        $L_{new} \leftarrow L_{new} \cup$ SEGMENTAUDIO(a_i, $P_{on} + \alpha_{on}, P_{off} + \alpha_{off}, T_{dur}$)
      else
        $L_{new} \leftarrow L_{new} \cup$ EQUALSEGMENT(a_i)
      end if
    end if
  end for
  return L_{new}
end function

Similar to the previous evaluation, this year's evaluation data are segmented using an automatic tool, which does not ensure that segments are proper sentences nor that they are aligned with the translated text. In addition, there is an apparent mismatch for segmentation between using voice activity detection (VAD) and segmenting by punctuations, where the latter is usually used for segmenting the training data. These assign extra importance to develop methods for proper segmentation of the audio data, which was confirmed in the previous year's evaluation campaign, where all top submissions used their own segmentation algorithm (Anastasopoulos et al., 2021).

Therefore, we design a segmentation algorithm based on a VAD model provided by pyannote.audio[9] (Bredin et al., 2020), as illustrated in Algorithm 1. We find that long segments are difficult for the model to decode and need to be further segmented. More specifically, we firstly use the VAD model pre-trained on AMI dataset (Carletta, 2007) to segment the audio. Two hyperparameters, P_{on} and P_{off}, are set for the VAD model, which are the onset speaker activation threshold and offset speaker activation threshold, respectively. Then the segments longer than T_{dur} are further segmented by increasing P_{on} and P_{off} with α_{on} and α_{off} if P_{on} and P_{off} are smaller than 0.95. Otherwise, we segment the audio into several parts with the same length smaller than T_{dur}, as large activation thresholds may lead to incorrect segmentation. In our experiments, We use the default values of the pre-trained model for P_{on} and P_{off}, which are 0.481

[9]https://huggingface.co/pyannote/voice-activity-detection

and 0.810. respectively. For segmenting long audios, we set the T_{dur} to 43.75 seconds, α_{on} to 0.1, and α_{off} to 0.028.

Moreover, some short segments are generated by the VAD model according to our observations, which may be incomplete sentences and harm the performance of our ST model. Merging the short segments helps the ST model utilize the context information. So we follow the algorithm in (Inaguma et al., 2021) to merge the short segments after the segmentation.

3 End-to-End YiTrans ST System

Recent studies, such as SpeechT5 (Ao et al., 2021) and SLAM (Bapna et al., 2021), have shown that joint pre-training of speech and text can boost the performance of spoken language processing tasks, such as speech translation. This section will mainly introduce the model architecture of our end-to-end YiTrans system, and the proposed methods to pre-train and fine-tune the models.

3.1 Model Architecture

Our evaluation system is based on an encoder-decoder model with state-of-the-art Transformer architecture. Figure 1 shows the framework of our end-to-end speech translation model, which consists of a speech encoder, text encoder, and text decoder. We employ the relative positional encoding (Shaw et al., 2018) for both the encoder and decoder network.

The speech encoder network contains a convolutional feature encoder and a Transformer encoder. The convolutional feature encoder is a convolutional network for extracting feature from waveform, which has seven 512-channel layers with kernel widths [10,3,3,3,3,2,2] and strides [5,2,2,2,2,2,2]. The Transformer encoder has 24 layers with model dimension 1024, inner dimension 4096 and 16 attention heads. The text encoder and decoder contain 12 layers and have a similar architecture to the Transformer encoder, except that the text decoder includes the cross-attention and the masked self attention. We optionally add an adaptor between the speech encoder and text encoder, which is three one-dimensional convolution layers with stride 2.

3.2 Multi-Stage Unified-Modal Pre-Training

To leverage large amounts of speech and text data, we firstly initialize the speech encoder with the

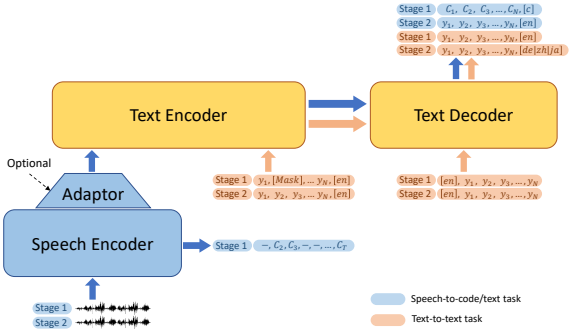

Figure 1: An illustration of the pre-training model.

HuBERT LARGE (Hsu et al., 2021) and the text encoder and decoder with the mBART50 (Tang et al., 2020). Then we design a multi-stage pre-training strategy to boost the performance of ASR and ST tasks.

In the first stage, we employ the speech to code pre-training method following Speech2C (Ao et al., 2022) to make full use of unlabeled speech data. More specifically, We set two pre-training tasks for the encoder-decoder pre-training using unlabeled speech data with pseudo codes, which are acoustic units learned from an offline clustering model. The encoder of Speech2C predicts the pseudo code via masked language modeling (MLM) in encoder output, like HuBERT model. In addition to MLM loss, the decoder of Speech2C learns to reconstruct pseudo codes auto-regressively, instead of generating real text transcription, both of which are discrete representations and have some semantic information corresponding to the speech signal. For the text data, the BART loss (Lewis et al., 2020) and cross entropy loss are used for the monolingual English data and MT data of three target languages, respectively. Note that the text data is only used for pre-training the text encoder and text decoder. For the second stage, we use the ASR data and the filtered MT data to continuously pre-train the model.

3.3 Joint Fine-Tuning

After pre-training, all the pre-trained modules (speech encoder, text encoder, text decoder and the optional adaptor) are used for directly fine-tunig a end-to-end ST model. We also make various efforts to improve the final perfermance.

Joint ST and MT Fine-Tuning We train the ST model along with an auxiliary text to text machine translation (MT) task. We utilize two methods from (Tang et al., 2021b) to enhance the performance of the primary ST task. First, a cross-attentive regu-

larization is introduced for the encoders. It minimizes the L2 distance between two reconstructed encoder output sequences and encourages the encoder outputs from different modalities to be closer to each other. Second, online knowledge distillation learning is introduced for MTL in order to enhance knowledge transfer from the MT to the ST task.

Synthetic Data for ST To provide more parallel audio-translation pairs, we translate the English side of the ASR data with our MT model. Specifically, we translate all the transcriptions of labeled ASR data listed in Table 1 to three target languages. For **en-de**, we additionally generate a certain amount of (about 8000 hours) cascaded pseudo data from unlabeled VoxPopuli, by firstly generating pseudo transcriptions with ASR model and then translating them with MT model.

Multi-Stage Fine-Tuning Note that our ST data is from various domains, including synthetic data and out-of-domain data (e.g. CoVoST). To make out ST model better adapted to the TED domain, we adopt the multi-stage fine-tuning method according to data category: At the first stage, we fine-tune ST models with all ST data, including synthetic and true data; Then at the second stage, the ST models are continually fine-tuned with in-domain data, i.e. Must-C and ST-TED.

3.4 Cascaded Speech Translation

To compare with our end-to-end YiTrans system, we also build a cascaded system by fine-tuning ASR and MT models from pre-trained models, and these subsystems also has been used to construct synthetic data for ST.

3.4.1 Automatic Speech Recognition

We fine-tune our ASR model with the following strategies: (1) **Synthetic Data for ASR**. To make the transcriptions contain the punctuations, we train a punctuation model using the English text of the MuST-C dataset, and add punctuations to the transcriptions of the TEDLIUM and LibriSpeech dataset with this model. We also use a model trained on MuST-C dataset to synthesize data from the Voxpopuli corpus. (2) **Data Filtering**. We find that the ASR data contains some noise and the transcription of some utterances are wrong. Therefore, we also use a model trained on MuST-C dataset to calculate the WER of each sentence, which is used for filtering ASR data. (3) **In-Domain Fine-Tuning**. To let the model fit the TED domain, we train two models from the second stage of pre-training. For the first one, we directly fine-tune the model on the MuST-C dataset. For the second one, we train the model with the TED-style datasets, which include MuST-C, ST-TED, and TED-LIUM corpus. We also filter the utterances that the WER is larger than 50% for the second model.

3.4.2 Machine Translation

All of our MT models for the offline task are fine-tuned from the big pre-trained mBART50 model, with advanced techniques: (1) We inherit the idea of **Progressive Learning** (Li et al., 2020) to train the model from shallow to deep. Specifically, our MT model has 24 encoders and 12 decoder layers, where the top 12 encoder layers are randomly initialized and the rest layers are initialized from mBART50. (2) **Back Translation**. Following previous experience in WMT evaluation campaigns (Akhbardeh et al., 2021), we use the trained {**de**,**ja**,**zh**}-**en** MT models to generate the English side for the selected monolingual text from Table 2. The MT models are also fine-tuned form mBART50. All back-translated pairs and the true paired data are combined for training. (3) **Multi-Stage Fine-Tuning**. We also perform multi-stage fine-tuning for MT models, where the model is first fine-tuned with all (processed) MT data, then is fine-tuned with in-domain data for a few steps. There is also an optional stage between them, which is fine-tuning with in-domain filtered data (the last line in Table 3). (4) **ASR Output Adaptation**. To alleviate the mismatch between the ASR transcripts and the real text used for training MT models, we add the synthetic in-domain data at the in-domain fine-tuning stage. The synthetic data is generated by replacing the English site text with pseudo ASR labels.

4 Experiments & Results

4.1 Pre-Training Setup

All models are implemented in Fairseq [10] (Ott et al., 2019). We pre-train two models depending on the computational efficiency. The first has 24 speech encoder layers, 12 text encoder layers and 12 decoder layers (denoted as PT48). The second has 12 encoder layers, an adaptor, 12 text encoder layers and 12 decoder layers (denoted as PT36). The total

[10] https://github.com/pytorch/fairseq

number of parameters for the pre-trained model is about 927M and 803M, respectively. The vocabulary size is 250k, which is inherited from the mBART50 model.

For the first stage, we pre-train our model on 64 A100 GPUs with a batch size of 37.5s samples per GPU for speech and 1875 tokens per GPU for text and set the update frequency to 3 for 100k steps. We optimize the model with Adam (Kingma and Ba, 2014) and set the learning rate to 3e-5, which is warmed up for the first 8% of updates and linearly decayed for the following updates. For the second stage, we also use 64 A100 GPUs and train the model for 300k with a batch size of 30s samples per GPU for speech and 1500 tokens for text. The learning rate set to 3e-5 is warmed up for the first 10% steps, held as a constant for the following 40% steps, and is decayed linearly for the rest steps. We add a language ID symbol for four languages at the start of each sentence.

ID	Model	tst2019	tst2020
1	Hubert & mBART	30.72	31.58
2	+ in-domain FT	30.62	33.07
3	PT36 + joint FT	20.10 (*)	20.12 (*)
4	+ in-domain FT	30.01	32.65
5	PT48	30.56	33.26
6	+ in-domain FT	30.98	33.48
7	+ joint FT	30.65	33.16
8	+ in-domain FT	**31.02**	33.46
9	+ cascaded data	31.00	**33.52**
10	+ in-domain FT	30.91	33.42
11	Ensemble (10, 6)	31.46	34.03
12	Ensemble (10, 8, 6)	31.49	33.84
13	Ensemble (10, 9, 8, 6)	31.47	33.95
14	Ensemble (10, 9, 8, 6, 2)	**31.57**	33.96
15	Ensemble (10, 9, 8, 6, 4, 2)	31.40	**34.10**

Table 4: BLEU results of e2e **en-de** models.

	Model	tst-common
1	Hubert & mBART	18.13
2	+ in-domain FT	18.59
3	PT36 + joint FT	18.16
4	+ in-domain FT	18.86
5	PT48	17.67
6	+ in-domain FT	18.30
7	+ joint FT	18.71
8	+ in-domain FT	**19.13**
9	Ensemble (8, 6)	19.38
10	Ensemble (8, 6, 2)	19.48
11	Ensemble (8, 6, 4)	19.70
12	Ensemble (8, 6, 4, 2)	**19.81**

Table 5: BLEU results of e2e **en-ja** models.

4.2 End-to-End Speech Translation

Our e2e ST models are fine-tuned from various pre-trained models. When fine-tuning with all ST data, the learning rate is set to 5e-5 and then is decayed linearly to zero within 200k training steps. And when fine-tuning with in-domain data, the learning rate is set to 1e-5 for 30k steps. All ST models are fine-tuned on 8 A100 GPUs with a batch size of about 30s per GPU and update frequency of 4.

	Model	tst-common
1	Hubert & mBART	28.69
2	+ in-domain FT	28.71
3	PT36	28.62
4	+ in-domain FT	28.61
5	PT48	29.07
6	+ in-domain FT	**29.26**
7	+ joint FT	28.51
8	+ in-domain FT	29.14
9	Ensemble (8, 6)	29.38
10	Ensemble (8, 6, 4)	29.36
11	Ensemble (8, 6, 2)	29.48
12	Ensemble (8, 6, 4, 2)	**29.53**

Table 6: BLEU results of e2e **en-zh** models.

en-de We use *tst2019* and *tst2020* as validation sets. We do not use *tst-common* as we find that it has overlapped speech samples with ST-TED training data. All BLEU results are computed at paragraph level, as listed in Table 4. It is noticed that almost all of the models get improved when fine-tuned with in-domain data (in-domain FT). What's more, joint ST&MT fine-tuning (joint FT) and adding cascaded pseudo ST data also help the performance. While, Table 4 shows that PT36 fine-tuned models get some unexpectedly bad results without in-domain fine-tuning. After checking the results we found that sometimes the model could only be able to decode a small portion of a sample especially when the sample is long. Finally, our PT48 fine-tuned model achieves the best performance, and ensemble decoding (Liu et al., 2018) with different models continually brings improvement. Our final submitted system is the last line of Table 4.

en-ja We use *tst-common* as the validation set, which has sentence-level translations so that BLEUs are computed at the sentence level. The results are listed in Table 5, where the BLEUs are computed after tokenized by Mecab[11]. Cascaded pseudo ST data is not performed due to the time urgency. Similar phenomena could be observed in Ta-

[11] https://taku910.github.io/mecab/

Model	en-de tst-common	en-ja/zh tst-common	tst2019	tst2020
Fine-tune with TED-Style data	8.49	8.67	10.9	13.4
Fine-tune with MuST-C	8.55	8.70	10.9	13.6
ensemble	8.47	8.56	10.7	13.3

Table 7: WER results of ASR Systems.

ble 5, where in-domain fine-tuning, joint ST&MT fine-tuning as well as model ensemble benefit the translation performance. Again, our PT48 fine-tuned model achieves the best performance. Our submitted system are listed in the last line of Table 5.

en-zh The validation set is also *tst-common* and sentence level BLEUs with character tokenizer are reported in Table 6. We find that in-domain fine-tuning and joint ST&MT fine-tuning are not as effective here as that in **en-de** and **en-ja**. That might be due to the specific data property of **en-zh**, e.g. all ST data is not mismatched very much with in-domain data. Finally, PT48 fine-tuned models still achieve the best performance and model ensemble brings improvement. Our final submitted system are listed in the last line of Table 6. Note that the results in Table 6 are not post-processed, while in our submitted results of *tst2022*, we post-process the decoding results by correcting the punctuation to Chinese style.

4.3 Cascade Speech Translation

Automatic Speech Recognition For the ASR fine-tuning, we use the CTC and cross-entropy loss to train the model (Watanabe et al., 2017). The loss weights are are set to 0.5 for both of them. We fine-tune the model on 8 A100 GPUs with the update frequency 4 for 120k steps, and set the batch size to around 30s samples per GPU. The learning rate set to 3e-5 is scheduled with the same strategy as the stage 2 of pre-training.

As shown in Table 10, we investigate the impact of speech segmentation with the model fine-tuned on MuST-C dataset. The pyannote toolkit improve the performance significantly compared to the given segmentation. The merge algorithm from Inaguma et al. (2021) further decreases the WER. We adjust two parameters of merge algorithm, M_{dur} and M_{int}. M_{dur} means the maximum duration after merging, and M_{int} is the minimum interval of two segments that will be merged. The experiments show that when M_{dur} and M_{int} are set to 30s and 1s, respectively, the model achieves the best performance. We then apply our Algorithm 1 to further segment the utterance longer than 43.75s, and the final WERs are 10.9 for tst2019 set and 13.6 for tst2020 set. Table 7 shows the WER scores of two ASR systems. We ensemble these two models and use the results for the cascade system.

Machine Translation For all three language pairs, we fine-tune both base models (with 12 encoder layers) and deep models (with 24 encoder layers) as described in Section 3.4.2. All models are fine-tuned on 8 A100 or V100 GPUs with a batch size of 2048 tokens per GPU, the update frequency is 1. The learning rate is set to 1e-4 with 5k warming up steps, then it is linearly decayed to zero in total 200k steps. In case of using additional back-translated data, we set the total training step to 300k. For in-domain fine-tuning, we only change the learning rate to 1e-5 and the total training step to 30k.

The results of MT systems are shown in Table 8. All BLEUs are computed the same way as e2e ST systems. Similar to e2e ST results, in-domain fine-tuning (in-domain FT) benefits all MT models. Progressive learning with deeper models also outperforms their baselines for all languages (line 3 vs. line 1). While, data filtering is shown effective for **en-de** but slightly negative for **en-zh**, which might because we remain too little data for **en-zh** to train such big models. It is also noticed that **en-ja** gets un-normal improvement from filtered data (indicated by *), we speculate data filtering might allow us to collect too similar text to *tst-common* to make the model overfit. Finally, back translation is shown benefit to all languages (line 7), while for **en-de** it falls slightly behind the best results, probably because of the amount of paired data already sufficient.

Cascade Systems Cascade systems are built upon ASR and MT systems. Table 9 shows the cascade ST results when applying the MT model

	Method	Model size	MT en-de tst-common	MT en-ja tst-common	MT en-zh tst-common
1	Baseline	12-12	35.82	19.58	28.52
2	+ in-domain FT	12-12	37.01	20.21	30.10
3	Deep model	24-12	36.25	20.15	29.19
4	+ data filtering	24-12	37.38	24.52 (*)	29.22
5	+ in-domain FT	24-12	**38.27**	**24.91** (*)	29.94
6	Back-translation	24-12	37.29	18.62	28.65
7	+ in-domain FT	24-12	38.05	20.92	**30.43**

Table 8: BLEU results of MT systems. * indicates the results may be over-fitted on tst-common set.

ID	Method	Model size	en-de tst-common	en-de tst2019	en-de tst2020	en-ja tst-common	en-zh tst-common
1	Baseline	12-12	33.07	30.47	32.96	18.79	27.50
2	+ in-domain FT	12-12	34.17	31.12	33.71	19.40	28.76
3	Deep model	24-12	33.29	30.67	33.14	19.00	27.81
4	+ data filtering	24-12	34.65	31.34	33.85	22.77 (*)	27.99
5	+ in-domain FT	24-12	**35.42**	31.63	**34.29**	**23.45** (*)	28.65
6	Back-translation	24-12	34.54	31.10	33.57	17.61	27.44
7	+ in-domain FT	24-12	35.40	**31.72**	34.16	19.94	**29.12**

Table 9: BLEU results of cascaded systems. * indicates the results may be over-fitted on tst-common set.

VAD	M_{dur}(s)	M_{int}(s)	tst2019	tst2020
Given	-	-	26.2	27.3
pyannote	-	-	15.7	16.3
	20	1	11.2	14.5
	25	0.5	12.4	15.0
	25	1	11.0	14.4
	25	1.5	11.6	14.3
	30	0.5	12.4	14.9
	30	1	10.9	14.0
	30	1.5	11.1	14.3
	35	1	11.4	14.0
Algo 1	30	1	**10.9**	**13.6**

Table 10: Comparison of segmentation ways and merge algorithm for ASR in terms of WER score.

Ensembled Models	tst-common	tst2019	tst2020
en-de			
MT #5; ST #10	**36.44**	**31.90**	**34.60**
MT #5,#7; ST #10	36.31	31.89	34.60
MT #5,#7,#4; ST #10	36.16	31.90	34.45
en-ja			
*MT #5; ST #8	22.79	\	\
*MT #5,#4; ST #8	**23.26**	\	\
*MT #5,#4,#7; ST #8	22.97	\	\
MT #7; ST #8	20.02	\	\
MT #7,#2; ST #8	20.12	\	\
MT #7,#2,#3; ST #8	**20.45**	\	\
en-zh			
MT #7; ST #6	29.38	\	\
MT #7,#2; ST #6	**29.48**	\	\
MT #7,#2,#5; ST #6	29.32	\	\

Table 11: BLEU results of cascaded systems. * indicates the results may be over-fitted on tst-common set.

listed in Table 8 to our best ASR systems. It is shown that better MT models always lead to better ST results. To leverage the end-to-end ST models, we also explore the ensemble of MT and end-to-end ST models as shown in Table 11. For **en-ja**, since the BLEU results of MT model #4 and #5 may be over-fitted on tst-common set, we also choose another three models for the ensemble.

5 Conclusion

In this paper we describe our End-to-End YiTrans speech translation system for IWSLT 2022 offline task. We explore building ST systems from large-scale pre-trained models. Our proposed multi-stage pre-training strategy allows the model to learn multi-modality information from both labeled and unlabeled data, which further improves the performance of downstream end-to-end ST tasks. Our systems are also built on several popular methods such as data augmentation, joint fine-tuning, model ensemble, and so on. Massive experiments demonstrate the effectiveness of our system, and show that the end-to-end YiTrans achieves comparable performance with the strong cascade systems and outperforms the last year's best end-to-end system by 5.2 BLEU in term of English-German tst2021 set.

References

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa,

Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. Findings of the 2021 conference on machine translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1–88, Online. Association for Computational Linguistics.

Antonios Anastasopoulos, Luisa Bentivogli, Marcely Z. Boito, Ondřej Bojar, Roldano Cattoni, Anna Currey, Georgiana Dinu, Kevin Duh, Maha Elbayad, Marcello Federico, Christian Federmann, Hongyu Gong, Roman Grundkiewicz, Barry Haddow, Benjamin Hsu, Dávid Javorský, Věra Kloudová, Surafel M. Lakew, Xutai Ma, Prashant Mathur, Paul McNamee, Kenton Murray, Maria Nădejde, Satoshi Nakamura, Matteo Negri, Jan Niehues, Xing Niu, Juan Pino, Elizabeth Salesky, Jiatong Shi, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Yogesh Virkar, Alex Waibel, Changhan Wang, and Shinji Watanabe. 2022. FINDINGS OF THE IWSLT 2022 EVALUATION CAMPAIGN. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, Dublin, Ireland. Association for Computational Linguistics.

Antonios Anastasopoulos, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alexander Waibel, Changhan Wang, and Matthew Wiesner. 2021. FINDINGS OF THE IWSLT 2021 EVALUATION CAMPAIGN. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 1–29, Bangkok, Thailand (online). Association for Computational Linguistics.

Junyi Ao, Rui Wang, Long Zhou, Shujie Liu, Shuo Ren, Yu Wu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, et al. 2021. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. *arXiv preprint arXiv:2110.07205*.

Junyi Ao, Ziqiang Zhang, Long Zhou, Shujie Liu, Haizhou Li, Tom Ko, Lirong Dai, Jinyu Li, Yao Qian, and Furu Wei. 2022. Pre-training transformer decoder for end-to-end asr model with unpaired speech data. *arXiv preprint arXiv:2203.17113*.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. 2019. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, volume 33, pages 12449–12460.

Ankur Bapna, Yu-an Chung, Nan Wu, Anmol Gulati, Ye Jia, Jonathan H Clark, Melvin Johnson, Jason Riesa, Alexis Conneau, and Yu Zhang. 2021. Slam: A unified encoder for speech and language modeling via speech-text joint pre-training. *arXiv preprint arXiv:2110.10329*.

Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. 2020. Pyannote.audio: Neural building blocks for speaker diarization. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7124–7128.

Jean Carletta. 2007. Unleashing the killer corpus: experiences in creating the multi-everything ami meeting corpus. *Language Resources and Evaluation*, 41:181–190.

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

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conference of european association for machine translation*, pages 261–268.

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.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Esteve. 2018. Tedlium 3: twice as much data and corpus repartition for experiments on speaker adaptation. In *International conference on speech and computer*, pages 198–208. Springer.

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.

Hirofumi Inaguma, Brian Yan, Siddharth Dalmia, Pengcheng Guo, Jiatong Shi, Kevin Duh, and Shinji Watanabe. 2021. ESPnet-ST IWSLT 2021 offline speech translation system. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 100–109, Bangkok, Thailand (online). Association for Computational Linguistics.

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 *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7180–7184. IEEE.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of machine translation summit x: papers*, pages 79–86.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Bei Li, Ziyang Wang, Hui Liu, Yufan Jiang, Quan Du, Tong Xiao, Huizhen Wang, and Jingbo Zhu. 2020. Shallow-to-deep training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 995–1005, Online. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles.

Yuchen Liu, Long Zhou, Yining Wang, Yang Zhao, Jiajun Zhang, and Chengqing Zong. 2018. A comparable study on model averaging, ensembling and reranking in nmt. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 299–308. Springer.

Jan Niehues, Rolando Cattoni, Sebastian Stüker, Mauro Cettolo, Marco Turchi, and Marcello Federico. 2018. The IWSLT 2018 evaluation campaign. In *Proceedings of the 15th International Conference on Spoken Language Translation*, pages 2–6, Brussels. International Conference on Spoken Language Translation.

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.

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.

Anthony Rousseau. 2013. Xenc: An open-source tool for data selection in natural language processing. *The Prague Bulletin of Mathematical Linguistics*, 100(1):73.

Anthony Rousseau, Paul Deléglise, and Yannick Estève. 2012. TED-LIUM: an automatic speech recognition dedicated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 125–129, Istanbul, Turkey. European Language Resources Association (ELRA).

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Yun Tang, Hongyu Gong, Xian Li, Changhan Wang, Juan Pino, Holger Schwenk, and Naman Goyal. 2021a. FST: the FAIR speech translation system for the IWSLT21 multilingual shared task. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 131–137, Bangkok, Thailand (online). Association for Computational Linguistics.

Yun Tang, Juan Pino, Xian Li, Changhan Wang, and Dmitriy Genzel. 2021b. Improving speech translation by understanding and learning from the auxiliary text translation task. *arXiv preprint arXiv:2107.05782*.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning.

Changhan Wang, Morgane Riviere, Ann Lee, Anne Wu, Chaitanya Talnikar, Daniel Haziza, Mary Williamson, Juan Pino, and Emmanuel Dupoux. 2021. VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. 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 993–1003, Online. Association for Computational Linguistics.

Changhan Wang, Anne Wu, and Juan Pino. 2020a. Covost 2: A massively multilingual speech-to-text translation corpus.

Qian Wang, Yuchen Liu, Cong Ma, Yu Lu, Yining Wang, Long Zhou, Yang Zhao, Jiajun Zhang, and Chengqing Zong. 2020b. CASIA's system for IWSLT 2020 open domain translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 130–139, Online. Association for Computational Linguistics.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. 2017. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

Jian Yang, Shuming Ma, Haoyang Huang, Dongdong Zhang, Li Dong, Shaohan Huang, Alexandre Muzio, Saksham Singhal, Hany Hassan Awadalla, Xia Song, et al. 2021. Multilingual machine translation systems from microsoft for wmt21 shared task. *arXiv preprint arXiv:2111.02086*.

A Appendix

We present the official test results for our submitted systems. For **en-de**, our end-to-end system achieves comparable performance with the cascade system, even the cascaded system is the ensemble of end-to-end and cascaded models. We also outperforms the best result of the last year by a great margin, especially for end-to-end systems. For **en-zh**, the gap between end-to-end and cascaded systems is also small (less than 1 point). While for **en-ja** cascaded systems performs better than end-to-end systems, probably because the end-to-end and cascaded models are complementary and resulting in a better ensemble. Meanwhile, it is noticed that adding punctuation for **en-ja** results is beneficial for *ref2* while harmful for *ref1*.

Model	BLEU ref2	BLEU ref1	BLEU both
Cascaded	25.6	23.7	36.4
E2E YiTrans	25.7	23.6	36.5

Table 12: Official results of our submitted **en-de** ST systems on tst2022.

Model	BLEU ref2	BLEU ref1	BLEU both
Cascaded			
IWSLT21 rank-1	24.6	20.3	34.0
The submission	28.1	23.2	39.0
End-to-end			
IWSLT21 rank-1	22.6	18.3	31.0
Our YiTrans	27.8	23.1	38.8

Table 13: Official results of our submitted **en-de** ST systems on tst2021.

Model	BLEU ref2	BLEU ref1	BLEU both
Cascaded	34.7	35.0	42.9
E2E YiTrans	34.1	34.6	42.3

Table 14: Official results of our submitted **en-zh** ST systems on tst2022.

Model	BLEU ref2	BLEU ref1	BLEU both
Cascaded	18.7	20.2	31.3
+ punc	22.8	14.7	30.0
E2E YiTrans	18.0	19.1	29.8
+ punc	21.8	13.7	28.2

Table 15: Official results of our submitted **en-ja** ST systems on tst2022.