USST's System for AutoSimTrans 2022

Jiahui Zhu [1] and **Jun Yu** [2]
[1] Shanghai University of Science and Technology
[2] East China University of Science and Technology
Shanghai, China
[1] zhujiahui.cn@outlook.com, [2] y45190321@mail.ecust.edu.cn

Abstract

This paper describes our submitted text-to-text Simultaneous translation (ST) system, which won the second place in the Chinese→English streaming translation task of AutoSimTrans 2022. Our baseline system is a BPE-based Transformer model trained with the PaddlePaddle framework. In our experiments, we employ data synthesis and ensemble approaches to enhance the base model. In order to bridge the gap between general domain and spoken domain, we select in-domain data from a general corpus and mix them with a spoken corpus for mixed fine-tuning. Finally, we adopt a fixed wait-k policy to transfer our full-sentence translation model to simultaneous translation model. Experiments on the development data show that our system outperforms the baseline system.

1 Introduction

Simultaneous translation (Gu et al., 2017; Ma et al., 2018) consists in generating a translation before the source speaker finishes speaking. It is widely used in many real-time scenarios such as international conferences, business negotiations and legal proceedings. The challenge of Simultaneous machine translation is to find a read-write policy that balances translation quality and latency. The translation quality will decline if the machine translation system reads insufficient source information. When reading wider source text, latency will increase.

Recent read-write policies can be divided into two categories: fixed policies such as wait-k (Ma et al., 2018), wait-if* (Cho and Esipova, 2016), and adaptive policies such as MoChA (Chiu and Raffel, 2017), MILk (Arivazhagan et al., 2019) and MU (Zhang et al., 2020). Fixed policies are simple to implement, but they neglect contextual information, which might result in quality reduction. Dynamic policies are more flexible, they can learn from data to achieve better quality/latency trade-offs, but accordingly difficult to train.

In our system, we train a Transformer (Vaswani et al., 2017) with a deep encoder (Meng et al., 2020) as baseline for abtaining rich source representations, besides we initialize the model with the method mentioned in DeepNet (Wang et al., 2022) in order to stabilize the training of the deeper model. At the pre-training stage, we firstly pretrain our model on a large general corpus, then we utilize data synthesis methods such as self-training and back-translation to improve model quality.

During the fine-tuning phase, we first apply fine-tuning on a small spoken corpus. For better domain adaptation, we adopt mixed fine-tuning (Chu et al., 2017), which trains on a mixed dataset that includes a subsampled general corpus and an upsampled spoken corpus. Thirdly, we propose a method called "in-domain mixed fine-tuning", which further improve the BLEU score than mixed fine-tuning. Specifically, inspired by in-domain data filtering (Moore and Lewis, 2010; Ng et al., 2019), we mixed upsampled spoken data with selected in-domain data from general corpus rather than random subsampled.

In the final stage, we employ the wait-k policy to convert the full-sentence translation model into a prefix-to-prefix architecture that predicts target words with only the source sentence's prefixes. After waiting for k-1 source subwords, the system reads a source subword and then predicts a target subword alternately until <eos> is detected. An example of wait 1 is shown in Figure 1.

The contributions of this paper are as follows:

- We propose a domain adaption approach called "in-domain mixed fine-tuning", which empirically proved to be better than fine-tuning while mitigating overfitting.
- All our code has been open sourced, see USST[1].

[1] https://github.com/tyy2022/USST_AutoSimultrans2022

Proceedings of the Third Workshop on Automatic Simultaneous Translation, pages 43 - 49

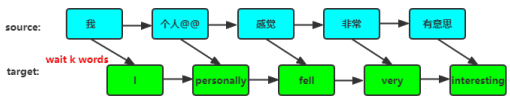

Figure 1: An example of prefix-to-prefix (wait 1).

2 Data

We participate in the Chinese-English streaming transcription track , where each sentence is broken into lines whose length is incremented by one word until the sentence is completed. An example is shown in Table 1.

Streaming transcription	Translation
我	
我下	I
我下面	
我下面来	'm
我下面来讲	
我下面来讲我	going
我下面来讲我们	
我下面来讲我们这	to
我下面来讲我们这段	talk
我下面来讲我们这段故	
我下面来讲我们这段故事	about
我下面来讲我们这段故事。	this story.

Table 1: An example of streaming input and output.

For pre-training, we use the CWMT21 parallel corpus (9.1M) [2], and we fine-tune the pre-trained model using transcription and translation of the BSTC (Baidu Speech Translation Corpus,37K) (Zhang et al., 2021), shown in Table 2. We also use CWMT's 10M Chinese monolingual data for synthetic data generation.

Similar to (Ng et al., 2019; Meng et al., 2020), we preprocess the data as follows:

- Word Segmentation: For Chinese, we use the open-source Chinese word segmentation tool jieba [3] for word segmentation. For English, we adopt punctuation-normalization, tokenization and truecasing with Moses scripts[4].
- Length filter: We remove sentences that are longer than 250 words and sentence pairs with a source/target length ratio exceeding 2.5.
- Langage identification (langid) (Lui and Baldwin, 2012): We use fastText[5] for language identification filtering, which removes sentence pairs that are not predicted as the correct language on either side.
- Deduplication: Remove duplicate sentences in Chinese monolingual data.
- Byte-pair-encoding (BPE) (Sennrich et al., 2016)[6]: For both the Chinese and English sides, we use BPE with 32K operations.

See Table 3 for details on the filtered data size.

Datasets	Domain	Train size	Dev size
CWMT21	General	9,023,708	1011
BSTC	Spoken	37,901	956

Table 2: Statistics of Chinese→English parallel corpus.

	Zh-En	Zh Mono
no filter	9.1M	10M
+length filter	8.9M	10M
+langid filter	8.8M	10M
+deduplication	-	6.8M

Table 3: Number of sentences in bitext and mono datasets for different filtering scheme

3 System Overview

3.1 Baseline System

As shown in previous work (Wang et al., 2019; Sun et al., 2019; Meng et al., 2020), increasing the depth of the Transformer encoder can substantially improve model performance, therefore we train the Transformer with deep encoder to obtain a better source representation.

In addition, in order to have both the high performance of post-norm and the stable training of pre-norm (Nguyen and Salazar, 2019), we use the methods mentioned in DeepNet (Wang et al., 2022), including a normalization function deepnorm that modifies the residual connection and a theoretically derived initialization. Our model configurations are shown in Table 4.

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

[3] https://github.com/fxsjy/jieba

[4] https://github.com/moses-smt/mosesdecoder

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

[6] https://github.com/rsennrich/subword-nmt

Configuration	Value
Encoder depth	12
Decoder depth	6
Attention heads	8
Embedding dim	512
FFN size	2048
Chinese vocab size	45942
English vocab size	32151
dropout	0.1

Table 4: Model Configuration

For training the full-sentence translation model, given the source sentence x, the probability of predicting the target sentence y is as shown in Eq. 1, and the training objective is to minimize the negative log-likelihood as shown in Eq. 2.

$$p(y|x) = \prod_{t=1}^{|y|} p(y_t|x, y_{<t}; \theta) \quad (1)$$

$$loss_{full}(\theta) = -\sum_{(x,y)\in D} logp_g(y|x;\theta) \quad (2)$$

The batch size for training is 4,096 tokens per GPU, and we trained our model for 7 epochs on 4 NVIDIA V100 GPUs for about 10 hours.

3.2 Data Synthesis

In order to improve the model performance, we used self-training and back-translation to synthesize pseudo-parallel corpus. Before using the two methods, we averaged 3 best checkpoints.

Self-training (He et al., 2019; Chen et al., 2020) uses a source-to-target model to generate synthetic pairs from source-side monolingual data to augment the original parallel corpus. We combined 2M Chinese monolingual data with 2M Chinese sentences randomly sampled from the CWMT parallel corpus, yielding a total of 4M monolingual for forward translation.

Reversely, back-translation (Sennrich et al., 2015; Edunov et al., 2018) first trains a target-to-source model, which then utilizes target-side monolingual data to synthesis a pseudo-parallel corpus. We randomly select 2M English sentences from the CWMT parallel corpus for back-translation.

We set the beam size to 5 for data generation, and then filtered out sentence pairs with normalized log score less than -3, resulting in a total of 5.7M pseudo-parallel sentences. Finally, we combined the pseudo corpus and the CWMT corpus to get a total of 14.5M sentences, and continued to train the forward model for 2 epochs, for about 2.5 hours on 4 V100 GPUSs.

3.3 Domain Adaption

A simple yet effective method for improving translation quality on the downstream task is fine-tuning with domain data,which is known as domain adaption (Luong and Manning, 2015). We train for another 2 epochs on the BSTC dataset with pre-trained model. Furthermore, we obverse that fine-tuning on limited spoken corpus lead to overfit quickly, as evidenced by the significant improvement on the BSTC development set while degrades rapidly on the CWMT development set.

In order to solve this issue, we explored mixed fine-tuning, an advanced domain adaption method that fine tunes a pre-trained model on a mixed corpus of in-domain and out-domain corpora. In addition, domain tags are added to all corpora to denote specific domains. In our experiments, we randomly sample 0.1M corpus from CWMT and upsample BSTC to 0.1M, then mix them up and shuffle randomly as for training set. For development set, we directly use the development set of BSTC rather than mixing in-domain and out-of-domain development sets.

We also verified the domain tags' efficacy and placement, the results show that appending the domain tags to source sentence performs best. However, in simultaneous translation task, this is unacceptable since the prefix-to-prefix model will not see the tag at the beginning.

To address this problem, we identify a 0.1M subset of CWMT that is most similar to BSTC by in-domain data filtering, then mixed the subset with upsampled 0.1M BSTC data. On the one hand, mixing increases the amount of domain data. On the other hand, there is no need to add tags because the mixed data only contains spoken domain.

For in-domain data filtering, given an in-domain data I, in this case BSTC, and a non-domain specific data N, in this case CWMT, we want to find the subset N_I that is drawn from the same distribution as I. Using Bayes' rule, we can calculate the probability that sentences in N is drawn from N_I for any given sentences, as shown in Eq. 3.

$$P(N_I|s, N) = \frac{P(s|N_I)P(N_I|N)}{P(s|N)} \quad (3)$$

$$logP(N_I|s, N) = logP(s|I) - logP(s|N) \quad (4)$$

Because I and N_I are drawn from the same distribution, we use $P(s|I)$ instead of $P(s|N_I)$. Besides, we neglect the $P(N_I|N)$ term because it will be constant for any given I and N.

Equivalently, in the log domain, the score of a sentence can be calculated as Eq. 4. This is similar to working with the cross-entropy difference: $H_I(s) - H_N(s)$, where $H_I(s)$ and $H_N(s)$ are the length-normalized cross entropy scores for a sentence s according to language models L_I and L_N.

$$Score(p)_{abs} = |P_I(p) - P_N(p)| \quad (5)$$

$$Score(p)_{noabs} = P_I(p) - P_N(p) \quad (6)$$

For simplicity, in this paper, we replace the monolingual sentence s with the sentence pair p drawn from non-domain corpus, the n-gram language model with Neural Machine Translation (NMT) model, the cross-entropy difference with perplexity absolute difference, as shown in Eq. 5, where $P_N(p)$ and $P_I(p)$ are the perplexity scores for a sentence pair p using an non-domain NMT_N model (pre-trained on CWMT) and an in-domain NMT_I model (fine-tuned on BSTC), respectively. We extracted 2M corpus from CWMT to calculate the absolute difference of perplexity scores, and screened 0.1M sentence pairs with the lowest scores, or about 5% of extracted data. We also tried to use the perplexity difference (see Eq. 6).

3.4 Ensemble

Averaging checkpoints is an easy but powerful ensemble method. We performed in-domain mix fine-tuning twice with two different random seeds, each taking the highest BLEU score checkpoint on the BSTC development set (up to 0.6 BLEU improvements).

System	BLEU	AL
pre-train	19.04	24.38
FT	25.11	24.35
In MF (abs)	26.35	24.33
+ensemble	**26.96**	24.34

Table 5: BLEU and Average Lagging on BSTC dev set. ("MF": Mixed fine-tuning)

3.5 Wait-k

The wait-k policy (Ma et al., 2018) refers to write target word y_t after reading source-side prefix $(x_1..x_{t+k-1})$. Let $g(t)$ be a monotonic non-decreasing function of t that indicates the number of source words read by the encoder when writing the target word y_t. Unlike full-sentence translation, the wait-k policy uses the source prefix $(x_1, \ldots, x_{g(t)})$ rather than the whole sentence x to generate y_t: $p(y_t|x_{\leq g(t)}, y_{<t})$. Thus, the decoding probability is shown in Eq. 7, and given training data D, the training objective is shown in Eq. 8.

$$p_g(y|x) = \prod_{t=1}^{|y|} p(y_t|x_{\leq g(t)}, y_{<t}; \theta) \quad (7)$$

$$loss_g(\theta) = -\sum_{(x,y)\in D} logp_g(y|x;\theta) \quad (8)$$

For k=1,3,5,7 we train 600 steps, and 300 steps for k=9 on the BSTC training set. Training more steps causes reduction of BLEU on the BSTC development set.

4 Experiments

Our system is implemented with the PaddlePaddle[7] framework, and our experiments are carried out on AI Studio[8] with 4 NVIDIA V100 GPU each of which has 32 GB memory. (We also benchmarked our code against fairseq [9] , see Appendix A)

4.1 Settings

For all experiments, we use the Adam optimizer with $\beta 1$ = 0.9, $\beta 2$ = 0.98. The initial learning rate is 1e-7, grows linearly to peak, then decayed proportionally to the inverse square root of the step number. During the training phase, we set peak learning rate to 5e-4, warmup step= 4000, max tokens= 4096, and update frequency = 4. Label smoothing with 0.1 is also adopted. The specific training parameters are shown in Table 7. We set beam size to 5 and length penalty to 1 during decoding.

4.2 Post-processing

For the post-processing after wait-k decoding, we apply de-truecaseing and de-tokenizing on the English translations with the scripts given in Moses.

[7] https://github.com/PaddlePaddle

[8] https://aistudio.baidu.com/aistudio/index

[9] https://github.com/facebookresearch/fairseq

System	CWMT Dev	BSTC Dev
Pre-train	27.39	16.93
+Fine-tuning	22.24	**20.46**
Self-training + Back-translation	**28.77**	16.55
+Fine-tuning	22.14	20.13
MF w/o tags	24.54	20.13
MF train tags(→)	24.74	19.23
MF train/test tags(→)	24.46	**20.86**
MF train tags(←)	**24.99**	19.60
MF train/test tags(←)	24.57	20.31
In MF (abs)	24.47	21.47
+ ensemble	**24.91**	**21.75**
In MF (no abs)	24.05	21.14

Table 6: Translation quality of our Chinese→English system. ("MF": Mixed fine-tuning; "w/o tags": With out tags; "train tags": Only add tags to the training set; "train/test tags": Add tags to both the training and test set; "→/←": Refers to whether to append or prepend tags to source text; "In MF": In-domain Mixed fine-tuning.)

Parameter	Pre-train	Fine-tune	Wait-k
Learning rate	5e-4	5e-5	5e-5
Warmup step	4000	500	500
Max tokens	4096	4096	512
Update frequency	4	1	1
Training time	7e	2e	600s

Table 7: Traing parameters. ("e": Epoch number; "s": Step number.)

4.3 Evaluation Metric

We use BLEU (Papineni et al., 2002) [10] and Average Lagging (AL) (Ma et al., 2018) [11] to evaluate translation quality and latency respectively. AL measures the degree the user is out of sync with the speaker. As shown in Eq.9-10, t is decoding step, τ is cut-off decoding step where source sentence is finished, $g(t)$ denotes the number of source words read by the encoder at decoding step t, and $r = |x|/|y|$ is the target-to-source length ratio. The smaller the AL (roughly equivalent to k) is, the more real-time the simultaneous translation system is.

$$AL_g(x, y) = \frac{1}{\tau} \sum_{t=1}^{\tau} g(t) - \frac{t-1}{r} \quad (9)$$

$$where\ \tau_g(|x|) = \min\{t|g(t) = |x|\} \quad (10)$$

4.4 Results and Analysis

Table 6 shows the translation quality variation of our system on the validation sets of CWMT and BSTC. The fine-tuning resulted in a significant improvement of 3.5 BLEU on BSTC, while dropping rapidly on CWMT with 5.1 BLEU. We observe that although using self-training and back-translation improves CWMT by 1.3 BLEU, it decreases by 0.4 BLEU on BSTC. This may be overfitting on the general domain and further deviating from the spoken domain. So in the later experiments, we directly use the pre-trained model to continue fine-tuning.

Lines 5–9 depict the mixed fine-tuning discussed in Section 3.3. We experimented with whether and where to add a tag, and discovered that adding tag at the end of source text works best, which is in line with the original paper's conclusion. The mixed fine-tuning reached 20.86 BLEU, a 0.4 BLEU improvement over the fine-tuning.

Finally, the in-domain mixed fine-tuning proposed in this paper is 0.6 BLEU better than mixed-fine-tuning, and after averaging two checkpoints, it further improved by 0.3 points to 21.75 BLEU, which is 1.3 BLEU higher than fine-tuning. In addition, we attempted to select the in-domain data using the perplexity difference (last row of Table 6, corresponding to Eq. 6), but the experimental re-

[10] https://dataset-bj.cdn.bcebos.com/qianyan/AST_Challenge.zip

[11] https://github.com/autosimtrans/SimulTransBaseline/blob/master/latency.py

sults proved to be less effective than the absolute value of the perplexity difference.

Table 5 and Figure 2 illustrates the translation quality and latency results after wait-k training. We set k=1, 3, 5, 7, 9, and train 600 steps until the training perplexity starts to rise. We only plot the results of using ensemble checkpoint for wait-k training since the effect of using in-domain mixed fine-tuning does not significantly exceed fine-tune when using wait-k training.

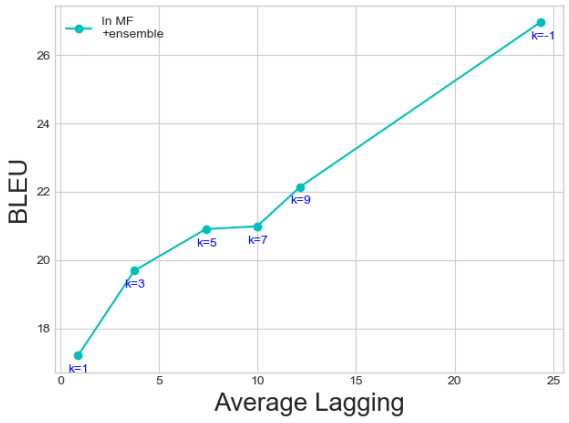

Figure 2: Translation quality (BLEU) against latency metric (AL) on Chinese→English (BSTC) simultaneous translation, showing the results of wait-k and full-sentences (k=-1) of the offline system. "In MF +ensemble" means using averaged checkpoints of the In-domain Mixed fine-tuning to perform wait-k training.

5 Conclusion

In this paper we describe our Chinese-to-English simultaneous translation system, which uses a deep Transformer to improve translation quality and adopts wait-k policy (Ma et al., 2018) to reduce latency. Besides, for better domain adaption, we combined mixed fine-tuning (Chu et al., 2017) with in-domain data filtering (Moore and Lewis, 2010; Ng et al., 2019) and proposed a new domain adaption method called "in-domain mixed fine-tuning", which is empirically more effective than fine-tuning and mixed fine-tuning.

In our future work, we plan to validate the effective of our proposed in-domain mixed fine-tuning on more datasets, while investigating some novel domain adaption methods. We also plan to research on some dynamic read-write policies in order to better balance quality and latency for simultaneous translation tasks.

Acknowledgements

This work was supported by Baidu's AI Studio, thanks to its free computing resources, and we sincerely hope that the PaddlePaddle framework will be built better. We also thank all the anonymous reviewers for their insightful and valuable comments.

References

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. *arXiv preprint arXiv:1906.05218*.

Peng-Jen Chen, Ann Lee, Changhan Wang, Naman Goyal, Angela Fan, Mary Williamson, and Jiatao Gu. 2020. Facebook ai's wmt20 news translation task submission. *arXiv preprint arXiv:2011.08298*.

Chung-Cheng Chiu and Colin Raffel. 2017. Monotonic chunkwise attention. *arXiv preprint arXiv:1712.05382*.

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

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

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.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2019. Revisiting self-training for neural sequence generation. *arXiv preprint arXiv:1909.13788*.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, et al. 2018. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. *arXiv preprint arXiv:1810.08398*.

Fandong Meng, Jianhao Yan, Yijin Liu, Yuan Gao, Xianfeng Zeng, Qinsong Zeng, Peng Li, Ming Chen, Jie Zhou, Sifan Liu, et al. 2020. Wechat neural machine translation systems for wmt20. *arXiv preprint arXiv:2010.00247*.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pages 220–224.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. *arXiv preprint arXiv:1907.06616*.

Toan Q Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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

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

Meng Sun, Bojian Jiang, Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Baidu neural machine translation systems for WMT19. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 374–381, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. 2022. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy. 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. *arXiv preprint arXiv:2104.03575*.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020. Learning adaptive segmentation policy for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2280–2289.

A Appendix: Benchmarking comparison of paddle and fairseq

We subsampled the CWMT dataset to 2M size, set the same parameters and then trained 20 epochs with fairseq and paddle's Transformer respectively, and the experimental results are as Table 8.

Codebase	Architecture	BLEU
Fairseq	base	23.08
Paddle	base	**23.18**
Paddle	base+deepnorm	23.15
Paddle	12+6+deepnorm	23.12

Table 8: A benchmark comparison of Transformers with different architecture implemented using paddle and fairseq.

where "base" is the Transformer base; "deepnorm" means using the initialization and residual connection modification methods in DeepNet, and the default initialization is the same as fairseq. "12+6" means 12-layer encoder and 6-layer decoder, which is used in this paper.

We observed that the Paddle version of the Transformer performed slightly better the fairseq version. Aside from that, the Transformer base seems to outperform our implementation of deepnorm, probably due to the size of the dataset. We will test it on a larger dataset in the future.