The RoyalFlush System for the WMT 2022 Efficiency Task

Bo Qin[1], **Aixin Jia**[1], **Qiang Wang**[2,1],
Jianning Lu[1], **Shuqin Pan**[1], **Haibo Wang**[1], **Ming Chen**[1*]
[1]RoyalFlush AI Research Institute, Hangzhou, China
[2]Zhejiang University, Hangzhou, China
{qinbo, jiaaixin, wangqiang3}@myhexin.com
{lujianning, panshuqin, wanghaibo3, chenming}@myhexin.com

Abstract

This paper describes the submission of the ROYALFLUSH neural machine translation system for the WMT 2022 translation efficiency task. Unlike the commonly used autoregressive translation system, we adopted a two-stage translation paradigm called Hybrid Regression Translation (HRT) to combine the advantages of autoregressive and non-autoregressive translation. Specifically, HRT first autoregressively generates a discontinuous sequence (e.g., make a prediction every k tokens, $k > 1$) and then fills in all previously skipped tokens at once in a non-autoregressive manner. Thus, we can easily trade off the translation quality and speed by adjusting k. In addition, by integrating other modeling techniques (e.g., sequence-level knowledge distillation and deep-encoder-shallow-decoder layer allocation strategy) and a mass of engineering efforts, HRT improves 80% inference speed and achieves equivalent translation performance with the same-capacity AT counterpart. Our fastest system reaches 6k+ words/second on the GPU latency setting, estimated to be about 3.1x faster than the last year's winner.

1 Introduction

Large-scale transformer models have made impressive progress in past WMT translation tasks, but it is still challenging for practical model deployment due to time-consuming inference speed (Wang et al., 2018b; Li et al., 2019). To build a fast and accurate machine translation system, participants in past WMT efficiency tasks developed and validated many efficient techniques, such as knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016), light network architecture (Kasai et al., 2020), quantization (Lin et al., 2020) etc. We noticed that all the above efforts are aimed at autoregressive translation (AT) models. In contrast, other translation paradigms, like non-autoregressive translation (NAT) (Gu et al., 2017) or semi-autoregressive translation (SAT) (Wang et al., 2018a) etc., have not been well studied.

In this participation, we restrict ourselves to the GPU latency track and attempt to investigate the potential of non-standard translation paradigms. However, replicating the vanilla non-autoregressive or semi-autoregressive models degrades the translation quality severely in our preliminary experiments. To this end, we explore hybrid-regressive translation (HRT), the two-stage translation prototype, to better combine the advantages of autoregressive and non-autoregressive translation (Wang et al., 2021b). Specifically, HRT first uses an autoregressive decoder to generate a discontinuous target sequence with the interval k ($k > 1$). Then, HRT fills the remaining slots at once with a non-autoregressive decoder. The two decoders share the same parameters without adding additional ones. Thus, HRT can easily trade-off between translation quality and speed by adjusting k [1]. Please see Table 1 for the comparison between different translation paradigms.

In addition to the change of translation paradigm, we have also made a mass of other optimizations. We use the widely used sequence-level knowledge distillation (Kim and Rush, 2016) and deep-encoder-shallow-decoder layer allocation strategy (Kasai et al., 2020) to learn effective compact models. Moreover, on the engineering side, we customized an efficient implementation of GPU memory reuse and kernel fusion for HRT following LightSeq (Wang et al., 2021c).

Putting all the efforts together, our HRT model achieves almost equivalent BLEU scores to the corresponding AT counterparts while improving the inference speed by about 80%. Our best-BLEU

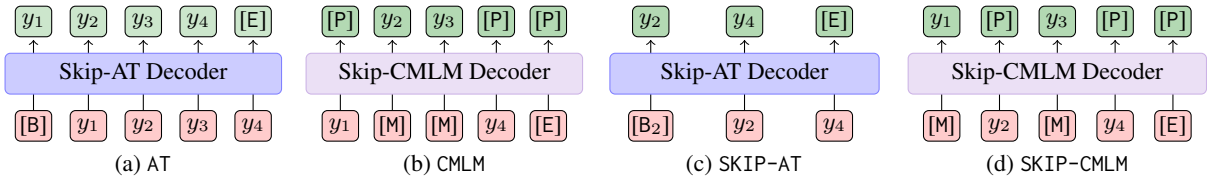
*Corresponding author.

[1]A larger k implies that fewer autoregressive decoding steps are required, resulting in faster inference speed but lower translation quality.

Proceedings of the Seventh Conference on Machine Translation (WMT), pages 671 - 676

Source	__The __Next __Big __Labor __ Strike __Hit s __Oregon
AT	__Der → __nächste → __große → __Arbeits → streik → __trifft → __Oregon → [EOS]
SAT	__Der __nächste → __große __Arbeits → streik __trifft → __Oregon [EOS]
NAT	**__Der __nächste __große __Arbeits streik __trifft __Oregon [EOS]**
HRT (Stage I)	__nächste → __Arbeits → __trifft → [EOS]
HRT (Stage II)	**__Der** __nächste **__große** __Arbeits **streik** __trifft **__Oregon** [EOS]

Table 1: Illustrations of different translation paradigms. __ is the special symbol for whitespace in sentencepiece. → denotes an autoregressive decoding step. **Blue** denotes that the token is generated in non-autoregressive way.



(a) AT (b) CMLM (c) SKIP-AT (d) SKIP-CMLM

Figure 1: Examples of training samples for four tasks, in which (a) and (b) are auxiliary tasks and (c) and (d) are primary tasks. For the sake of clarity, we omit the source sequence. [B]/[E]/[P]/[M] represents the special token for [BOS]/[EOS]/[PAD]/[MASK], respectively. [B_2] is the [BOS] for k=2. Loss at [P] is ignored.

model drops an average of 0.9 BLEU points compared to the teacher model, which ensembles four transformer-big models. Moreover, our fastest model decodes 6k+ source words per second, estimated to be 3.1x faster than the winner in last year [2].

2 Hybrid-regressive translation

One of the most important highlights is the introduction of the newly proposed two-stage translation prototype——HRT. In this section, we will detail the model, training, and decoding of HRT.

2.1 Model

HRT consists of three components: encoder, Skip-AT decoder (for stage I), and Skip-CMLM decoder (for stage II). All components adopt the Transformer architecture (Vaswani et al., 2017). The two decoders have the same network structure, and we share them to make the parameter size of HRT the same as the vanilla Transformer. The only difference between the two decoders lies in the masking pattern in self-attention: The Skip-AT decoder masks future tokens to guarantee strict left-to-right generation like an autoregressive Transformer (Vaswani et al., 2017). In contrast, the Skip-CMLM decoder eliminates it to leverage the bi-directional context like the standard conditional masked language model (CMLM) (Ghazvininejad et al., 2019). We note that there is no specific target length prediction module in HRT because HRT can obtain the translation length as the by-product of the Skip-AT decoder: N_{nat}=$k \times N_{at}$, where N_{at} is the sequence length produced by Skip-AT.

2.2 Training

Multi-task framework. We learn HRT through joint training of four tasks, including two primary tasks (SKIP-AT, SKIP-CMLM) and two auxiliary tasks (AT, CMLM). All tasks use cross-entropy as the training objective. Figure 1 illustrates the differences in training samples among these tasks. It should be noted that, compared with AT, SKIP-AT shrinks the sequence length from N to N/k, whereas the token positions follow the original sequence. For example, in Figure 1 (c), the position of Skip-AT input ([B_2], y_2, y_4) is (0, 2, 4) instead of (0, 1, 2). Involving auxiliary tasks is necessary because the two primary tasks cannot fully leverage all tokens in the sequence due to the fixed k. For example, in Figure 1 (c) and (d), y_1 and y_3 have no chance to be learned as the decoder input of either SKIP-AT or SKIP-CMLM.

Curriculum learning. To ensure that the model is not overly biased towards auxiliary tasks, we propose gradually transferring the training tasks from

[2]We obtained the best decoding speed in last year's competition according to Heafield et al. (2021): The fastest system *2.12_1.micro.rowcol-0.5* decodes 19,951,184 space-separated words in 13665 seconds. Therefore, we estimated its inference speed is 1460 words/second. We note that the acceleration ratio is not an accurate value because we use slightly different computation devices and test data to measure the speed.

auxiliary tasks to primary tasks through curriculum learning (Bengio et al., 2009). More concretely, given a batch of original sentence pairs $\mathcal{B}$, and the proportion of primary tasks in $\mathcal{B}$ is p_k, we start with p_k=0 and construct the training samples of AT and CMLM for all pairs. Then we gradually increase p_k to introduce more learning signals for SKIP-AT and SKIP-CMLM until p_k=1. In implementation, we schedule p_k by:

$$p_k = (t/T)^{\lambda}, \quad (1)$$

where t and T are the current and total training steps. λ is a hyperparameter, and we use λ=1 to increase p_k linearly for all experiments.

2.3 Decoding

HRT adopts two-stage generation strategy: In the first stage, the Skip-AT decoder starts from [BOS$_k$] to autoregressively generate a discontinuous target sequence $\hat{\boldsymbol{y}}_{at} = (z_1, z_2, \ldots, z_m)$ with chunk size k until meeting [EOS]. Then we construct the input of Skip-CMLM decoder $\boldsymbol{y}_{nat}$ by appending $k-1$ [MASK]s before every z_i. The final translation is generated by replacing all [MASK]s with the predicted tokens by the Skip-CMLM decoder with one iteration. If there are multiple [EOS]s existing, we truncate to the first [EOS]. Note that the beam size b_{at} in Skip-AT can be different from the beam size b_{nat} in Skip-CMLM as long as st. $\mathrm{b}_{at} \geq \mathrm{b}_{nat}$: We only feed the top b_{nat} Skip-AT hypothesis to Skip-CMLM decoder. Finally, we choose the translation hypothesis with the highest score $\mathrm{S}(\hat{\mathrm{y}})$ by:

$$\underbrace{\sum_{i=1}^{m} \log P(z_i|\boldsymbol{x}, \boldsymbol{z}_{<i})}_{\text{Skip-AT score}} + \underbrace{\sum_{i=0}^{m-1}\sum_{j=1}^{k-1} \log P(\hat{y}_{i\times k+j}|\boldsymbol{x}, \boldsymbol{y}_{nat})}_{\text{Skip-CMLM score}} \quad (2)$$

where z_i=$\hat{y}_{i\times k}$.

3 Optimization

Sequence-level knowledge distillation. Overall, we use the teacher-student framework via sequence-level knowledge distillation (SEQKD) to learn our small HRT model (Kim and Rush, 2016). Specifically, the ensemble of provided four transformer-big models is our teacher, whose beam search results are used as our distillation data. There are 320M official distillation data composed of 80M parallel and 240M monolingual datasets. We directly use the distillation data without further data cleaning. We use the same sentencepiece vocabulary as the teacher model to encode the text.

Encoder	Newstest19	Newstest20	WPS
6	43.8	32.6	4.0k
12	45.6	33.9	3.8k
20	45.9	34.4	3.5k

Table 2: SacreBLEU and inference speed against the number of encoder layers in HRT with a single-layer decoder. All HRT models are trained with k=2. WPS refers to source words per second, measured by the average five runs with a batch size of 1. Unless otherwise stated, we measure WPS on Newstest20.

b_{at}	b_{nat}	Newstest19	Speedup
5	5	**45.9**	ref.
5	1	45.8	1.05x
1	1	45.6	**1.33x**

Table 3: Effects of different settings of beam size in HRT.

Deep-encoder-shallow-decoder architecture. Using deep-encoder-shallow-decoder network architecture has been widely validated effectiveness for transformer-based NMT systems (Wang et al., 2021a; Kasai et al., 2020). Our HRT also follows this guidance by using only one decoder layer. We use the pre-norm transformer following Wang et al. (2019) to learn deep encoder well. Intuitively, the single-layer decoder may be insufficient for HRT because the decoder is responsible for both autoregressive and non-autoregressive generation. However, as shown in Table 2, we found that HRT enjoys the deep-encoder-shallow-decoder architecture. For example, compared to HRT_E6D1 [3], HRT_E12D1 and HRT_E20D1 improve +1.6/+2.0 BLEU score points on average, while the inference speed decreases by 5% and 12.5%. Therefore, we mainly investigate HRT with a 12-layer and 20-layer encoder due to the high BLEU scores.

Fully greedy search. Prior work has validated that greedy search is sufficient for the autoregressive distilled model to work well (Kim and Rush, 2016). Since HRT refers to two beam sizes (b_{at} and b_{nat}), we test three settings as shown in Table 3. It can be seen that using b_{at}=1 and b_{nat}=1 only decreases BLEU slightly but accelerates a 30%+ faster than that of b_{at}=5 and b_{nat}=5. Unless otherwise stated, we use b_{at}=1 and b_{nat}=1 in the following experiments.

[3] We use HRT_E{#1}D{#2} to denote the HRT model with {#1}-layer encoder and {#2}-layer decoder.

Model	Param.	Newstest19		Newstest20		Average		WPS
		BLEU	COMET	BLEU	COMET	BLEU	COMET	
Teacher (four transformer-big)	4×209.1M	47.1	-	35.0	-	41.1	-	-
WMT21 fastest (Behnke et al., 2021)	9.0M	-	-	33.3	-	-	-	1.5k*
AT_E6D1	39.5M	45.1	0.551	33.9	0.469	39.5	0.510	2.2k
AT_E12D1	58.4M	45.4	0.572	34.1	0.489	39.8	0.530	2.1k
AT_E20D1	83.6M	45.9	0.581	34.6	0.502	40.3	0.541	1.9k
HRT_E12D1 (k=2)	58.4M	45.6	0.547	33.9	0.454	39.8	0.500	3.8k
HRT_E12D1 (k=3)	58.4M	45.0	0.503	33.6	0.377	39.3	0.440	4.9k
HRT_E12D1 (k=4)	58.4M	44.1	0.432	32.9	0.267	38.5	0.350	**6.1k**
HRT_E20D1 (k=2)	83.6M	45.9	0.561	34.4	0.472	**40.2**	**0.517**	3.5k
HRT_E20D1 (k=3)	83.6M	45.3	0.524	34.0	0.406	39.7	0.465	4.5k
HRT_E20D1 (k=4)	83.6M	44.2	0.435	33.3	0.283	38.8	0.360	5.4k

Table 4: Compare different model variants with regard to SacreBLEU (Post, 2018), COMET (Rei et al., 2020) and inference speed. $*$ denotes the number is not exactly comparable due to the difference in test data and GPU.

Maximum sequence length. We predefine the maximum source/target sequence length L as 200. Here the length is calculated based on the results of sentencepiece. Once the sequence length exceeds L, we truncate the source/target sequence. For HRT, we let the maximum decoding length in the Skip-AT stage as L/k. In this way, the maximum target length in the Skip-CMLM stage can be guaranteed not beyond L.

GPU memory reuse. Since we only participate in the GPU latency track, given the predefined maximum sequence length L, we can estimate the maximum GPU memory buffer used in the encoder, autoregressive decoder, and non-autoregressive decoder in advance, respectively. Then we only allocate the maximum buffer size among them because these three processes are memory-independent. This memory-reuse method helps us reduce our footprints and avoid frequent memory applications and releases.

Kernel fusion. Too many fine-grained kernel functions make modern GPU inefficient due to kernel launching overhead and frequent memory I/O addressing (Wang et al., 2021c; Wu et al., 2021). We follow the good implementation in LightSeq and use the general matrix multiply (GEMM) provided by cuBLAS as much as possible, with some custom kernel functions. Please refer to Wang et al. (2021c) for details.

FP16 inference. We also use the 16-bit floating-point to utilize modern GPU hardware efficiently. Previous study (Wang et al., 2021a) shows that FP16 can bring significant acceleration in batch decoding. In contrast, in our GPU latency task, we only observed about 15% speedup due to the smaller computational burden, as shown in Table 5.

Model	FP16	WPS
HRT_E12D1 (k=2)	no	3.3k
HRT_E12D1 (k=2)	yes	3.8k

Table 5: The effect of FP16 on HRT model in GPU latency task.

Docker submission. We use multistage builds to reduce the docker image size. Specifically, we first use static compilation to build our executable program with CUDA 11.2. Then we add the built result and model into the 11.2.0-base-centos7 docker. The model disk size is compressed by *xz* compression toolkit.

4 Experimental Results

Setup. We mainly compared HRT to the standard autoregressive baselines in Table 4. All models adopt transformer-base setting (Vaswani et al., 2017): d=512, d_{ff}=2048, $head$=8. We validated the following model variants:

- **AT:** We train three autoregressive baselines with the number of encoder layers of 6, 12, and 20, denoted as AT_E6D1, AT_E12D1, and AT_E20D1, respectively. All AT models are trained from scratch for 300k steps.
- **HRT:** HRT models are fine-tuned based on the pre-trained AT counterparts for 300k steps. We also try different chunk sizes $k \in \{2, 3, 4\}$ to trade off the translation quality and inference speed.

Other training hyper-parameters are the same as Wang et al. (2019). We ran all experiments on 8 GeForce 3090 GPUs. For decoding, the length penalty is 0.6, and the batch size is 1. We report the detokenized SacreBLEU score with the same signature as the teacher. Besides, we also follow Helcl et al. (2022)'s suggestion to provide COMET score (Rei et al., 2020) for the evaluation of non-autoregressive translation.

Translation quality. First, we can see that a deeper encoder improves about 0.5 BLEU points across the board. When using k=2 for HRT, both 12-layer and 20-layer HRT models have almost equivalent BLEU scores to that of AT counterparts. Our best HRT model HRT_E20D1 with k=2 only drops an average of 0.9 BLEU points than the teacher using model ensemble. However, in line with Helcl et al. (2022), we find that even when BLEU scores are close, HRT's COMET scores are significantly lower than those of AT, e.g., AT_E20D1 vs. HRT_E20D1 (k=2). Nevertheless, HRT_E20D1 (k=2) still achieves higher BLEU and COMET than AT_E6D1 with 60% acceleration.

Translation speed. We estimated the inference speed of the fastest system last year according to the data in Heafield et al. (2021). Supposing ignoring the difference in test data, our AT baselines run about 40%+ faster than it. It indicates that our AT engine is a strong baseline. Even so, we can see that both 12-layer and 20-layer HRT with k=2 achieve approximated 80% acceleration than AT without BLEU drop. Moreover, larger k further reduces the autoregressive decoding steps: Our fastest model, HRT_E12D1 (k=4), decodes 6k+ source words/second, which is 3.1 times faster than the fastest system last year.

5 Conclusion

This paper presented the ROYALFLUSH system to the GPU latency track of the WMT 2022 translation efficiency task. We proposed hybrid-regressive translation, a novel two-stage prototype to replace conventional autoregressive translation. With a lot of development optimization, we showed that our HRT with a chunk size of 2 achieves equivalent translation performance to the AT counterpart while accelerating 80% inference speed. By increasing HRT's chunk size, our system can further speed up 60% to 6k+ words/second, estimated to be about 3.1 times faster than the fastest system in last year's competition.

References

Maximiliana Behnke, Nikolay Bogoychev, Alham Fikri Aji, Kenneth Heafield, Graeme Nail, Qianqian Zhu, Svetlana Tchistiakova, Jelmer van der Linde, Pinzhen Chen, Sidharth Kashyap, and Roman Grundkiewicz. 2021. Efficient machine translation with model pruning and quantization. In *Proceedings of the Sixth Conference on Machine Translation*, pages 775–780, Online. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6111–6120, Hong Kong, China. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Kenneth Heafield, Qianqian Zhu, and Roman Grundkiewicz. 2021. Findings of the WMT 2021 shared task on efficient translation. In *Proceedings of the Sixth Conference on Machine Translation*, pages 639–651, Online. Association for Computational Linguistics.

Jindřich Helcl, Barry Haddow, and Alexandra Birch. 2022. Non-autoregressive machine translation: It's not as fast as it seems. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1780–1790, Seattle, United States. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2020. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. *arXiv preprint arXiv:2006.10369*.

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.

Bei Li, Yinqiao Li, Chen Xu, Ye Lin, Jiqiang Liu, Hui Liu, Ziyang Wang, Yuhao Zhang, Nuo Xu, Zeyang Wang, Kai Feng, Hexuan Chen, Tengbo Liu, Yanyang Li, Qiang Wang, Tong Xiao, and Jingbo Zhu. 2019. The NiuTrans machine translation systems for WMT19. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 257–266, Florence, Italy. Association for Computational Linguistics.

Ye Lin, Yanyang Li, Tengbo Liu, Tong Xiao, Tongran Liu, and Jingbo Zhu. 2020. Towards fully 8-bit integer inference for the transformer model. *ArXiv*, abs/2009.08034.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. 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. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Chenglong Wang, Chi Hu, Yongyu Mu, Zhongxiang Yan, Siming Wu, Yimin Hu, Hang Cao, Bei Li, Ye Lin, Tong Xiao, and Jingbo Zhu. 2021a. The NiuTrans system for the WMT 2021 efficiency task. In *Proceedings of the Sixth Conference on Machine Translation*, pages 787–794, Online. Association for Computational Linguistics.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018a. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488.

Qiang Wang, Bei Li, Jiqiang Liu, Bojian Jiang, Zheyang Zhang, Yinqiao Li, Ye Lin, Tong Xiao, and Jingbo Zhu. 2018b. The NiuTrans machine translation system for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 528–534, Belgium, Brussels. Association for Computational Linguistics.

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.

Qiang Wang, Heng Yu, Shaohui Kuang, and Weihua Luo. 2021b. Hybrid-regressive neural machine translation.

Xiaohui Wang, Ying Xiong, Yang Wei, Mingxuan Wang, and Lei Li. 2021c. LightSeq: A high performance inference library for transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pages 113–120, Online. Association for Computational Linguistics.

Kaixin Wu, Bojie Hu, and Qi Ju. 2021. TenTrans high-performance inference toolkit for WMT2021 efficiency task. In *Proceedings of the Sixth Conference on Machine Translation*, pages 795–798, Online. Association for Computational Linguistics.