HW-TSC's Submissions to the WMT22 Efficiency Task

Hengchao Shang[1], **Ting Hu**[2], **Daimeng Wei**[1], **Zongyao Li**[1],
Xianzhi Yu[2], **Jianfei Feng**[2], **Jinlong Yang**[1], **Zhiqiang Rao**[1], **Ting Zhu**[1],
Zhengzhe Yu[1], **Lizhi Lei**[1], **Shimin Tao**[1], **Hao Yang**[1], **Ying Qin**[1]
[1]Huawei Translation Service Center, Beijing, China
[2]Huawei Noah's Ark Lab, Hong Kong, China
{shanghengchao,huting35,weidaimeng,lizongyao,yuxianzhi,
fengjianfei1,yangjinlong7,raozhiqiang,zhuting20,
yuzhengzhe,leilizhi,taoshimin,yanghao30,qinying}@huawei.com

Abstract

This paper presents the submissions of Huawei Translation Services Center (HW-TSC) to WMT 2022 Efficiency Shared Task. For this year's task, we still apply sentence-level distillation strategy to train small models with different configurations. Then, we integrate the average attention mechanism into the lightweight RNN model to pursue more efficient decoding. We add a retrain step to our 8-bit and 4-bit models to achieve a balance between model size and translation quality. We still use Huawei Noah's Bolt[1] for INT8 inference and 4-bit storage. With Bolt's support for batch inference and multi-core parallel computing, we finally submit models with different configurations to the CPU latency and throughput tracks to explore the Pareto frontiers.

1 Introduction

Transformer and its variants (Vaswani et al., 2017; Shaw et al., 2018; So et al., 2019; Dehghani et al., 2019) have become benchmark models for machine translation. A lot of innovations and engineering optimizations (Tay et al., 2020) in this area are based on Transformer. However, with the increase of bilingual and monolingual data sizes used for training, the size of the model expands and the requirement of computing ability become higher. Taking T5 (Raffel et al., 2020), GPT3 (Brown et al., 2020) and a series of subsequent large models (Fedus et al., 2021; Smith et al., 2022) as examples, although they have achieved very good performances, it is still difficult for ordinary practitioners to reproduce or use these models for research and industry application. Especially in scenarios where hardware capability is limited, models that balance size, quality and power consumption is urgently needed. The WMT Efficiency task is performed under such constraints.

[1]https://github.com/huawei-noah/bolt

In this year's task, we still focus on CPU inference optimization and participate in CPU latency and multi-core throughput tracks.

We employ knowledge distillation (Hinton et al., 2015) to train small models. The teacher models and distillation data come from official website. We only perform simple data cleaning, and all of our experiments are conducted based on fariseq (Ott et al., 2019).

Deep encoder and shallow deocder models can balance quality and inference speed (Wang et al., 2019). We follow this configuration for pursuing extreme efficient decoding. Inspired by SRU++ (Lei, 2021) and AAN (Zhang et al., 2018), we integrate the average attention mechanism with a lightweight RNN for more efficient decoding.

We retrain our 8-bit quantisation model (Jacob et al., 2018), then compare its result with that of direct post-quantization (Sung et al., 2015) model. We finally find that in the distillation scenario, the difference between the two is not obvious. We apply 4-bit storage to obtain an extremely small model size. Although our training and inference strategies ensure basically the same model quality, the gap in overall quality is large and the model needs to be further optimized.

We still use Huawei Noah's Bolt as the inference library. This year, we implement batch inference and parallel computing on multi-core CPUs for the throughput track.

Finally, after performing some necessary engineering optimizations, we submit four models with different configurations to explore the Pareto frontiers.

2 Teacher to Student Knowledge Distillation

2.1 Data Process

The task is to translate English to German following the constrained news task from WMT 2021.

Proceedings of the Seventh Conference on Machine Translation (WMT), pages 677–681
Abu Dhabi, December 7–8, 2022.

The teacher model, as well as bilingual data and distillation data used in this task are provided by the organizer. It makes everyone on the same start line in the distillation experiment, avoiding the quality difference due to different teacher models. We download the data and find it pretty much the same as the data we used in the task last year. According to our distillation experiment last year, keeping the ratio of bilingual data and distillation data as 1:2 can ensure that the student model inherits the knowledge of the teacher model well. Except for the generation of distillation data, other processing strategies are the same as last year's. For details, please refer to our previous task report (Shang et al., 2021).

2.2 Vocabulary

We build a joint subword segmentation model from real parallel data using SentencePiece (Kudo and Richardson, 2018) as last year. The vocabulary size is set to 25k tokens.

2.3 Model Structure

The autoregressive module is based on the self-attention in the Transformer decoder layer. The decoding complexity increases as the decoding length increases. Therefore, special processing is required if we want to pursue extreme decoding performance. The commonly used strategy is to replace it with a fixed computational cost module, such as LSTM, other RNN variants (Lei et al., 2018), or AAN. These modules use a global cell to store sentence-level information and perform the same cell update actions as each token is decoded without relying on the decoded sequence.

SRU++ (Lei, 2021) further replaces the heavy-weight multiplication operation outside the cell with a self-attention component to improve the representation ability of the model. We use the AAN to replace the standard self-attention module for faster decoding while ensuring the expression ability of the model. We call it the AASRU model.

The calculation formula in the cell is as follows:

$$
\begin{aligned}
f[t] &= \sigma(U[t,0]) + V \odot C[t-1] + b) \\
r[t] &= \sigma(U[t,1]) + V^{'} \odot C[t-1] + b^{'}) \\
c[t] &= f[t] \odot C[t-1] + (1 - f[t]) \odot C[t,2] \\
h[t] &= r[t] \odot C[t] + (1 - r[t]) \odot x[t]
\end{aligned}
$$

The formula for calculating U is as follows:

$$
\begin{aligned}
Q &= W^{q} X^{T} \\
V &= W^{v} X^{T} \\
A^{T} &= AVERAGE(V^{T}) \\
U^{T} &= W^{o} layernorm(Q + A)
\end{aligned}
$$

where W^{q} and $W^{v} \in R^{d^{'} \times d}$, $W^{o} \in R^{3d \times d^{'}}$, d is the hidden state size, and $d^{'}$ is the attention dimension. The σ is the sigmode function, and $\odot$ is the element-wise multiplication, t refer to the time step, v and b are parameter vectors to be learnt during training, c and h are the cell states and the hidden states in RNN.

2.4 Training

Our distillation experiments are based on fairseq. We implement the AASRU module by referring to the open-source transformer-aan [2]. Also, we do not use regularization techniques such as dropout and label smoothing. All our models are trained using 8 Nvidia Tesla V100 for about two days. The maximum number of tokens vary from 4096 to 10240 according to the model size, as we try to keep the maximum GPU memory usage the same.

After that, we retrain our 8-bit quantization model, constrain all Linear and Matual operator's inputs to the interval [-1, 1], add quantization and inverse quantization operators to the model graph. The retrain is performed after the base model has been trained for 200K steps.

We compare the results of retrain and post-quantization on the Base.12 model, and find almost no difference in performances of the two models under the current distillation experiment setting. Therefore, we submit the post-quantized models.

Next, we apply 4-bit storage models to pursue extreme model sizes. In order to achieve better translation, we add retrain and verify the consistency between training and inference. The translation quality obtained via Bolt inference and training respectively is almost the same. However, the overall quality of our model declines greatly, requiring further optimization.

2.5 Evaluation

We still use WMT 2019 and 2020 News Task test sets to measure our models with SacreBLEU (Post, 2018) this year. We perform a simple post-processing (normalize the punctuation) on the German translations, so the BLEU scores are slightly higher than the officially provided one.

[2] https://github.com/bzhangGo/transformer-aan

Model	Emb.	FFN	Head	Depth	Params(M)	Size(MB)	wmt19	wmt20
Teacher*4	1024	4096	16	6/6	200	800	47.08	36.29
Base.12	512	2048	8	12/1	53	210	45.75	35.30
Base.12 + 8-bit	512	2048	8	12/1	53	210	45.89	35.20
Base.6	512	2048	8	6/1	35	140	44.78	34.59
Small.12	384	1536	6	12/1	33	132	45.03	34.89
Small.9	384	1536	6	9/1	28	112	44.62	34.40
Small.6	384	1536	6	6/1	22	88	43.80	34.26
Tiny.12	256	1024	4	12/1	17	68	43.84	33.62
Tiny.6	256	1024	4	6/1	13	52	42.15	32.27
Tiny.6 + 4-bit	256	1024	4	6/1	13	52	34.75	26.30

Table 1: Results of Distillation Training. 8-bit and 4-bit refer to retraining.

Overall, the results of our distillation experiments are within our expectations. The Baseline model has about 25% parameters as the teacher model, and its performance is attenuated by about 1.5 BLEU. The 8-bit retraining model is basically the same as the direct training one. However, we observe over 5.0 BLEU decrease on our Tiny.6 model after adding 4-bit storage. The reason may be that we treat every parameter the same way, including embedding. As a result, more training tricks and experiments are required in the future.

We also analyze the effect of the encoder's height and width on the model. Comparing Base.6 to Small.12, and Small.6 to Tiny.12, we find that deeper networks almost have equal or better quality even with less parameters except for Tiny.12's 2020 test result.

Under the same height setting, models with different widths also perform differently. A wider model seems to perform better. Comparing 12-layer and 6-layer models, we observe less than 1 BLEU difference under the base setting, less than 1.2 BLEU difference under the small setting (and only 0.7 BLEU difference on the WMT20 test set), and only about 1.7 BLEU difference under the tiny setting. Wider encoder means more parameters and probably better quality.

Based on the above analysis and the quality gap between the models, We finally decide to submit four models including Base.12, Small.9, Tiny.12, and Tiny.6 to explore the Pareto frontiers better.

3 Inference Optimizations

We use Bolt acceleration library as CPU optimization backend to build the high-performance translation engine. Bolt has a standalone C++ runtime, therefore it can perform fast inference without any third-party dependencies. We use Bolt v1.4.0, which will be available in October 2022.

3.1 8-bit Quantization

We still apply the post-training quantization method this year. All parameters of the model except the bias are quantized to 8-bit intergers by absolute maximum quantization. All GEMM operations in the attention layer are in 8-bit and well optimized by Intel VNNI instructions, but the layernorm and softmax computations are back off to FP32.

3.2 4-bit Storage

For this year's submission, we employ 4-bit storage to achieve almost 8x model compression. With 4-bit storage, all parameters have to be converted to 8-bit integers for calculation because of the hardware limitations, so there is no performance advantage compared with 8-bit storage.

3.3 Batch and Thread

For the throughput track, we support batch inference and merge multiple matrix calculations in attention layer. Our experiments show an end-to-end speedup of up to 20% on a single core. To further increase the throughput, we divide the input text into specified sizes and assign them to multiple CPU cores for parallel computation. The input text is sorted first to prevent the performance waste due to the difference of data lengths within the batch. In the submitted systems, we uniformly set the batch size to 4.

3.4 Other Strategies

We apply some other commonly used strategies such as greedy decoding, caching and shortlist,

Model	Precious	Size	WPS	BLEU
Teacher	FP32	2000	-	36.29
Base.12	FP32	212	237	35.30
	INT8	53	815	35.20
+retrain	INT8	53	815	35.19
Small.9	FP32	112	468	34.40
	INT8	28	1129	34.29
Tiny.12	FP32	68	759	33.62
	INT8	17	1693	33.39
Tiny.6	FP32	52	996	32.27
	INT8	13	2001	31.92
+int4	INT8	6.5	1989	26.10

Table 2: Optimization results. The test set is WMT 2020 News test. The unit of size is MB. WPS refers to the source side. The test environment is Intel(R) Xeon(R) Gold 6278C CPU @ 2.60GH. We submit four models: Base.12, Small.9, Tiny.12 and Tiny.6 and the final Tiny.6+int4

which can improve the model decoding efficiency to a certain extent. Details can be found in our last year's report.

4 Optimization Results

Our final optimization results are shown in Table 2. We find that the inference speed of our models is significantly improved through int8 inference, and the overall improvement is 2-3 times that of FP32, which is basically the same as last year's results.

By analyzing the results of our comparative experiments on Base.12, we find that BLEU is only slightly decreased when we directly use the post-quantization inference version. So there is not much room left when optimizing the performance of models that employ retraining. The reason may be the limited diversity of the model under the distillation setting. The post-quantization model basically meet our requirement on quality.

We additionally employ 4-bit storage on the Tiny.6 model for pursuing extreme model size. After retraining, we successfully compress the model to almost 1/8 of the original size, and maintain a high degree of consistency between training (26.30 BLEU) and inference (26.10 BLEU) with slightly BLEU score decrease. We also submit the model for evaluation.

When preparing the model for the throughput track, we need to set the batch size for batch translation. We compare the impact of different batch sizes on throughput in detail using our Base.12 model. Results are shown in Table 3. When the batch size exceeds 3, the improvement becomes insignificant. Considering that the hardware used in the task may be different from our test environment, we set the batch size to 4 for all of our submissions for convenience.

Batch Size	BLEU	Costs	WPS
Base.12	35.30	-	-
1	35.20	53	815
2	35.23	45	978
3	35.17	44	1000
4	35.22	44	1000
8	35.38	45	978
16	35.26	45	978

Table 3: The effect of batch size on throughput. WPS refers to the source side.

5 Submitted Docker Images

Due to the simple runtime environment of Bolt, we can choose a very basic image to run our system. We still apply the ubuntu:18.04. Our inference project is inherited from last year's, adding support of batch inference and using a thread pool to run models in parallel on multiple CPU cores. Following the task requirements, our startup script is /run.sh. Our model is stored in the /model directory, which contains the converted Bolt model, vocabulary, and shortlist files. The compressed file is provided.

Our largest model volume is around 50M, and the base image volume is around 60M. The space occupied by our inference project is almost negligible, so the final image we submitted after compression still does not exceed 70M, and the smallest one is about 35M.

6 Concolusion

In this year's task, we follow some strategies from last year, including data processing, basic distillation training, etc. In addition, we explore a new and more efficient decoding structure, AASRU, this year, which reduces the amount of computation while maintaining quality. We add 8-bit and 4-bit retrain to distillation training, and verify the consistency of training and inference. Regarding engineering, we add the relevant features of Batch inference and multi-core parallel computing, and finally submit several models with balanced quality and speed for CPU latency and throughput tracks.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2019. Universal transformers.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71.

Tao Lei. 2021. When attention meets fast recurrence: Training language models with reduced compute. *arXiv preprint arXiv:2102.12459*.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence.

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 NAACL-HLT 2019: Demonstrations*.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Hengchao Shang, Ting Hu, Daimeng Wei, Zongyao Li, Jianfei Feng, Zhengzhe Yu, Jiaxin Guo, Shaojun Li, Lizhi Lei, Shimin Tao, et al. 2021. Hw-tsc's participation in the wmt 2021 efficiency shared task. In *Proceedings of the Sixth Conference on Machine Translation*, pages 781–786.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

David R. So, Chen Liang, and Quoc V. Le. 2019. The evolved transformer.

Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. 2015. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

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*.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. *arXiv preprint arXiv:1805.00631*.