Prune Before Converged: Efficiently Training Pruned Transformer Models for Neural Machine Translation

Anonymous EMNLP submission

Abstract

The Transformer models have been proven effective in machine translation task. To deploy the huge-size Transformer models in real-world applications, pruning methods compress the models by dropping out redundant attention heads. However, most of existing pruning methods only work for well-trained models, leading to massive redundant computations in the training stage. In this work, we propose a structured pruning method to efficiently train Transformer models for neural machine translation. We construct experiments on two widely-used machine translation datasets, and results show that pruning early before convergence significantly saves total training time while keeps comparable performance.

1 Introduction

The architecture of Transformer (Vaswani et al., 2017) has shown its tremendous power and achieved great success in machine translation task (Brown et al., 1993; Och, 2003; Sutskever et al., 2014; Bahdanau et al., 2014). The Transformer models, however, have grown increasingly huge (Devlin et al., 2018; Radford et al.; Yang et al., 2019; Raffel et al., 2019; Brown et al., 2020), resulting in substantial computational cost and high inference latency. This prevents them being deployed in resource-limited and real-time applications, such as online services and edge devices.

A mainstream approach to model compression is known as pruning. Unstructured pruning (weight pruning) (Han et al., 2015; Gordon et al., 2020) unrestrictedly removes the redundant parameters in Transformer models, but its pruned sparse matrices usually gain almost no acceleration on common hardware (Wen et al., 2016), and thus cannot effectively speed up the model despite significant model size reduction. On the other hand, Structured pruning (Kovaleva et al., 2019; Voita et al., 2019; Michel et al., 2019) finds that a considerable number of the attention heads in the well-trained Transformer models are redundant. By pruning these redundant heads, the Transformer models could achieve significant inference speedup while maintaining their performance.

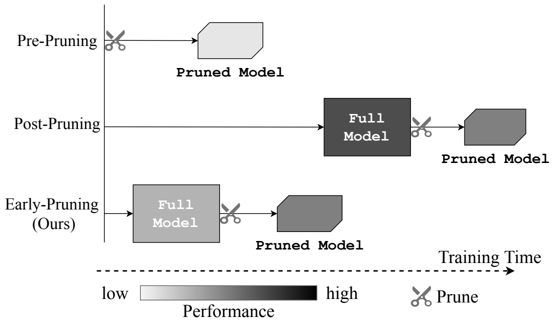

Figure 1: Comparison of different pruning strategies. We propose to prune early in the training stage to save training time while maintain the performance.

A major drawback of pruning methods is that it can only work for fully-trained models. However, the huge cost in training large Transformer models also greatly limits its utilization in real-world applications. In addition, pruning methods can recover most performance of a large model, which directly training a small model can not (Voita et al., 2019). This raises an interesting question: can we prune the Transformer model early in the training stage?

In this work, we propose a structured pruning method to efficiently train Transformer models for neural machine translation (NMT). Instead of pruning on well-trained models, we propose to prune the model far before converged. We add trainable weights to detect redundant attention heads and feed-forward network (FFN) dimensions and prune them to reduce the model size early.

We evaluate our early pruning approach on datasets of two language pairs. Experiments show that the method of early pruning can keep comparable performance with pruning after convergence, while could save training time by more than 50%.

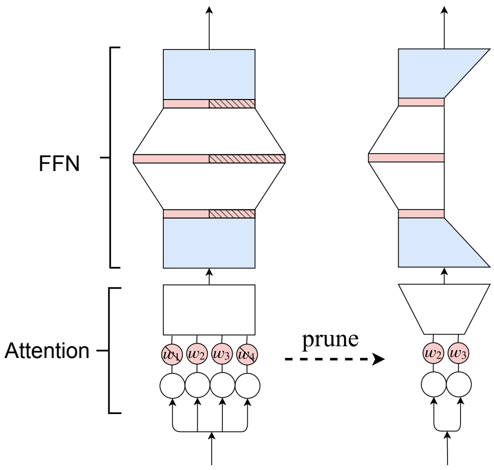

Figure 2: A brief view of a pruned Transformer layer. White circles are attention heads. Red components are additional weights. Blue parts are extra linear projections to keep dimension matched after pruning.

2 Related Work

Model compression works are mainly divided into three categories: (1) Knowledge distillation methods, which transfer the knowledge of the original trained "teacher" model to a lite one (Hinton et al., 2015; Jiao et al., 2019; Sanh et al., 2019; Sun et al., 2019, 2020; Jiao et al., 2020; Hou et al., 2020). (2) Quantization methods, which transform model weights in low precision formats to compress the model size (Shen et al., 2019; Zafrir et al., 2019; Han et al., 2016). (3) Pruning methods, which remove the redundant components in models (Voita et al., 2019; Michel et al., 2019; McCarley et al., 2019; Li et al., 2020; Wang et al., 2019; Zhang et al., 2020; Fan et al., 2019). Our work focuses on the pruning methods, while the main difference is that we prune the model far before convergence , saving much redundant computational costs during training.

Lottery Ticket Hypothesis (LTH) (Frankle and Carbin, 2018) shows that training carefully pruned model with re-initialization can recover the performance of full model. You et al. (2019) further shows that such good ticket can be determined early in the training stage. Recently, the existence of winning tickets is also verified in the field of NLP (Yu et al., 2019; Renda et al., 2020). Chen et al. (2020a,b); Prasanna et al. (2020) transfer LTH to Transformer based models. Our work provides a feasible way, i.e., early structured pruning, to find such a lucky ticket for Transformer models in neural machine translation.

3 Our Method

In this section we introduce the details of our early structured pruning method to efficiently train pruned Transformer models. We prune the models far before it converges (e.g., train with a few steps). The pruning strategy can be divided into two steps: (1) Prune the attention layer; (2) Prune the feed-forward layer according to the number of pruned attention heads for each layer.

3.1 Attention Layer Pruning

We first detect the importance of attention heads, and then prune unimportant ones. Inspired by Ahmed et al. (2017), we simply add a trainable weight w_i multiplied on each attention head to measure its importance. Thus, the modified Multi-Head Attention (MHA) can be formulated as:

$$\begin{aligned}
\mathrm{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \mathrm{Concat}(\mathbf{h}_1,\cdots,\mathbf{h}_\mathrm{n})\mathbf{W}^O \\
\mathbf{h}_\mathrm{i} &= f(w_i)\cdot \mathrm{Attn}(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V), \\
\mathrm{Attn}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}
\end{aligned} \tag{1}$$

where $f(\cdot)$ is a ReLU function to avoid negative weight. Trainable head weight w_i is shown in Figure 2 as red circles. During training, we first initialize each w_i to 1, and then optimize them using the objective $\mathcal{L}_{NMT}(\mathbf{w}_i)$ with Transformer's weights fixed, where $\mathcal{L}_{NMT}$ is the objective of NMT.

We prune attention heads with smaller head weights from encoder, decoder and cross attention by a pre-defined pruning ratio. Thus, the number of pruned heads in each layer may differ, indicating different importance of these layers. Please refer to Appendix A for further details of pruning.

3.2 Feed-Forward Layer Pruning

We suppose that after attention layer pruning, layers with less attention heads also do not require full dimension in its corresponding FFN. Similarly as attention layer, we add weights multiplied on each dimension of FFN layers (including input, intermediate and output layers) to detect redundancy. Different from attention layer pruning, we add two additional prunable linear projections before and after the original FFN to keep the dimension matched with attention module after pruning. The modified

FFN is formulated as:

$$
\begin{aligned}
\mathbf{h}_{in} &= (\mathbf{x}\mathbf{W}_{entry} + \mathbf{b}_{entry}) \circ f(\mathbf{w}_{in}) \\
\mathbf{h}_{inter} &= f(\mathbf{h}_{in}\mathbf{W}_1 + \mathbf{b}_1) \circ f(\mathbf{w}_{inter}) \\
\mathbf{h}_{out} &= (\mathbf{h}_{inter}\mathbf{W}_2 + \mathbf{b}_2) \circ f(\mathbf{w}_{out}) \\
\mathbf{o} &= \mathbf{h}_{out}\mathbf{W}_{exit} + \mathbf{b}_{exit}
\end{aligned}, \quad (2)
$$

where $\mathbf{W}_{entry}, \mathbf{b}_{entry}, \mathbf{W}_{exit}, \mathbf{b}_{exit}$ are the parameters of two additional entry and exit projections, as shown in Figure 2 as blue rectangles. "$\circ$" stands for dimension-wise multiplication. $\mathbf{w}_{in}, \mathbf{w}_{inter}, \mathbf{w}_{out}$ are weights on input, intermediate and output layers, as shown in Figure 2 as red rectangles. We call them FFN weights in brief. During training, we first initialize two additional projections as identity functions and FFN weights as 1, and then optimize FFN weights using the objective $\mathcal{L}_{NMT}(\mathbf{w}_{in}, \mathbf{w}_{inter}, \mathbf{w}_{out})$ similarly.

We prune the dimension for each FFN layer by the ratio of attention heads pruned in that layer.[1] For example, if half attention heads are pruned, half of FFN dimensions should be pruned (shown in Figure 2 as shadowed part of red rectangles). Please refer to Appendix A for further pruning details.

We find that a large proportion of parameters of Transformer models lies in FFN layers, so pruning FFN layers is very crucial for efficient training and inference. As show in Figure 3, by pruning FFN layers along with attention heads, the size of Transformer model can be significantly reduced, and its inference computations (measured by FLOPs) are also greatly saved compared with pruning attention heads alone.

4 Experiments

4.1 Setup

We evaluate our proposed method on WMT14 English-German (En-De) and WMT17 Chinese-English (Zh-En) translation tasks. The evaluation metric is case-sensitive BLEU. We implement our method on top of an open-source machine translation toolkit THUMT (Tan et al., 2020). Please refer to Appendix B for detailed experimental settings.

Training Time Comparing We directly compare training time in the same settings. All runs in our pruning experiments contain three stages: train full model with additional parts (colored component in Figure 2 left) disabled, which is equivalent to a standard Transformer; activate additional parts, train weights and prune the model; train the pruned model. To better observe the convergence time of the final pruned model, we extend the training process of the pruned model to be long enough and observe the training curve of it. We compare following settings with notations:

Pre-Pruning Prune the model at the beginning. This is equivalent to training a smaller pruned model from scratch.

Post-Pruning Prune the model after convergence. This is conventional practice and serves as the baseline setting here.

Early-Pruning Prune the model early in the training stage before convergence. This is what we propose.

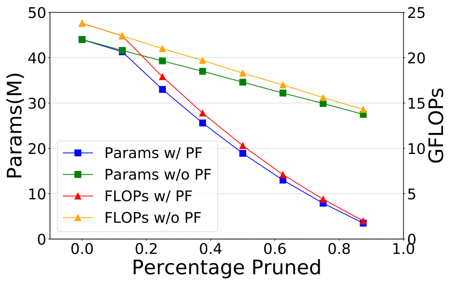

Figure 3: The influence of pruning percentage on model size and FLOPs. "PF" refers to Pruning FFN layers. Parameters and FLOPs are computed theoretically without embedding layer.

4.2 Results

Time Efficiency We conduct experiments on both En-De and Zh-En tasks. The results are listed in Table 1. The "Transformer" line represents training a full Transformer model until convergence, which is 100k steps for base model and 200k steps for big model. The pruning ratio is fixed to 30% in all runs, and pruning time step in Early-Prune is step 30k. We compare the training curves of big models on En-De dataset in Figure 4. From the curves, we can observe that Early-Pruning converges in less than 40h, while Post-Pruning costs about 80h to converge. Our method converges earlier by more than 50%. This conclusion remains true on other settings and datasets, please refer to Appendix C.2 for curves of other settings and datasets. According to Table 1, the performance of Post-Pruning and Early-Pruning are comparable in both sizes of model and both language pairs, indicating that

[1]In particular, we only prune the decoder FFN layer when heads in cross attention are pruned. Since we don't prune FFN layer when heads in decoder self-attention are pruned, the model size after pruning may differ given same pruning ratio.

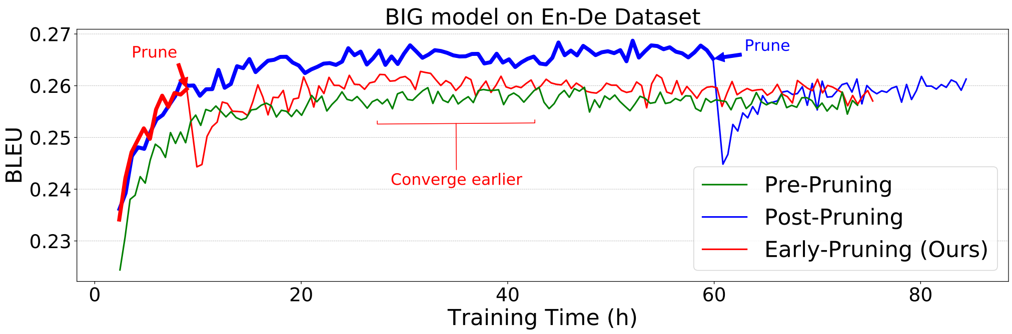

Figure 4: Training curves of big models on En-De dataset. BLEU scores are evaluated on the dev set. We annotate the pruning point on the curve. We train full model before the pruning point (thick line) and train pruned model after it (thin line).

	En-De		Zh-En	
	Params	BLEU	Params	BLEU
BASE model				
Transformer	98.7M	27.54	93.2M	23.6
Pre-Pruning	86.0M	26.66	79.9M	22.9
Post-Pruning	84.8M	27.15	77.0M	23.3
Early-Pruning (Ours)	83.8M	27.13	78.9M	23.2
BIG model				
Transformer	285.4M	29.01	274.5M	23.9
Pre-Pruning	228.0M	27.60	223.0M	23.4
Post-Pruning	223.2M	28.24	227.8M	23.6
Early-Pruning (Ours)	230.0M	28.14	222.9M	23.7

Table 1: Results on WMT14 En-De task and WMT17 Zh-En task.

Pruning Ratio	Post-Pruning	Early-Pruning (Ours)
20%	27.32	27.31
50%	26.65	26.50
80%	24.19	24.33

Table 2: Comparing results when change prune ratio. We prune the model at step 30k. The results are evaluated on WMT14 En-De dataset.

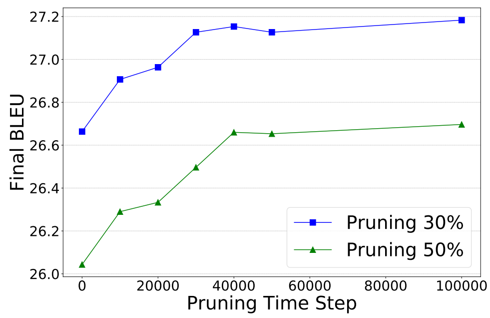

Figure 5: Influence of different pruning time step on final performance.

pruning before convergence doesn't harm the final performance of the pruned model. Comparing with Pre-Pruning, our Early-Pruning approach achieves much better performance, proving the necessity of training full model before pruning.

Influence of Pruning Ratio We compare with Post-Pruning with a pruning ratio of 20%, 50% and 80%, as shown in Table 2. From the table, we observe that the performance of Early-Pruning and Post-Pruning are comparable in various pruning ratios, verifying that the effectiveness of our early pruning approach is irrelevant to pruning ratio.

Influence of Pruning Time Step We also observe the influence of pruning time step with two different pruning ratios, as shown in Figure 5. There is a trend that the later we prune, the better final performance will be. But the increasing rate of the performance gradually declines, and there is a moment when pruning after it and before it behave differently. We infer from Figure 5 that the critical point is about step 30k for pruning 30% and 40k for pruning 50%. Pruning before it causes significant performance drop, while pruning after it makes slight difference in final performance. Since the critical point is quite a few steps before convergence, it gives evidence that pruning after the critical point before convergence could save training time while hardly harm the final performance.

5 Conclusion

In this work, we present an efficient training method of pruned Transformer models by pruning early in the training stage. With suitable pruning ratio and pruning time step, our method can save more than 50% training time with comparable performance. In the future, we will explore how to apply our model to pre-trained language models, and extend it into other model architectures.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted Transformer Network for Machine Translation. *arXiv preprint arXiv:1711.02132*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Peter E Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *arXiv preprint arXiv:2005.14165*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020a. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. *Advances in Neural Information Processing Systems*, 33.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2020b. Early-BERT: Efficient BERT Training via Early-bird Lottery Tickets. *arXiv preprint arXiv:2101.00063*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing Transformer Depth on Demand with Structured Dropout. *arXiv preprint arXiv:1909.11556*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. *arXiv preprint arXiv:2002.08307*.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both Weights and Connections for Efficient Neural Networks. *Advances in Neural Information Processing Systems*, 2015-January:1135–1143.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. DynaBERT: Dynamic BERT with Adaptive Width and Depth. *Advances in Neural Information Processing Systems*, 33.

Xiaoqi Jiao, Huating Chang, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Improving Task-Agnostic BERT Distillation with Layer Mapping Search. *arXiv preprint arXiv:2012.06153*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. TinyBERT: Distilling BERT for Natural Language Understanding. *arXiv preprint arXiv:1909.10351*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 4365–4374.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. 2020. Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers. *International Conference on Machine Learning*, pages 5958–5968.

J. S. McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structured Pruning of a BERT-based Question Answering Model. *arXiv preprint arXiv:1910.06360*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? *Advances in Neural Information Processing Systems*, 32:14014–14024.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. pages 160–167. Association for Computational Linguistics (ACL).

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, Belgium, Brussels. Association for Computational Linguistics.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT Plays the Lottery, All Tickets Are Winning. *arXiv preprint arXiv:2005.00561*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv preprint arXiv:1910.10683*, 21:1–67.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing Rewinding and Fine-tuning in Neural Network Pruning. *arXiv preprint arXiv:2003.02389*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv:1508.07909*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian May, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2019. Q-BERT: Hessian based ultra low precision quantization of BERT. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821. arXiv.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient Knowledge Distillation for BERT Model Compression. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 4323–4332.

Zewei Sun, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2020. Alleviating the Inequality of Attention Heads for Neural Machine Translation. *arXiv preprint arXiv:2009.09672*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhixing Tan, Jiacheng Zhang, Xuancheng Huang, Gang Chen, Shuo Wang, Huanbo Luan, and Yang Liu. 2020. THUMT: An Open-Source Toolkit for Neural Machine Translation. Technical report.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 5797–5808.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured Pruning of Large Language Models. *arXiv preprint arXiv:1910.04732*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2082–2090. Neural information processing systems foundation.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G. Baraniuk, Zhangyang Wang, and Yingyan Lin. 2019. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2019. Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP. *arXiv preprint arXiv:1906.02768*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: Quantized 8Bit BERT. *arXiv preprint arXiv:1910.06188*.

Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. 2020. Know What You Don't Need: Single-Shot Meta-Pruning for Attention Heads. *arXiv preprint arXiv:2011.03770*.

A Pruning Details

A.1 Attention Heads

When an attention head is decided to be pruned, we simply remove W_i^Q, W_i^K, W_i^V of the chosen head. The output projection W_O can be considered a concatenation of W_i^O matching each head. We prune the W_i^O of chosen head as well. As shown in Figure 2, the dimension of hidden state is reduced in pruned attention heads, while the dimension of input and output keeps unchanged.

A.2 FFN Dimensions

When a dimension in a hidden layer in the FFN is chosen to be pruned, the two linear projections related to the hidden layer are to be pruned. The chosen dimension is directly linked to a corresponding column in the linear projection before the hidden layer and a corresponding row in the linear projection after hidden layer. The linked row and column are pruned to reduce the model size. As shown in Figure 2, we add additional weights on three hidden layers in the FFN, thus four projections $\mathbf{W}_{entry}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_{exit}$ are pruned. If all heads are pruned in a particular layer, then all dimensions should be pruned. In practice we directly remove the whole layer.

B Experiment Setting

Model setting We perform pruning on base model and big model respectively. Both models contains 6 layers for encoder self-attention, decoder self-attention and cross attention. The hidden size of the base model is 512 and is split into 8 heads, while the hidden size of the big model is 1024 and is split into 16 heads.

Data preparing We conduct our experiments on two language pairs, English-German and Chinese-English. For English-German task, We trained our model on WMT 2014 (Bojar et al., 2014) English-German dataset. The training set contains 4.5 M sentence pairs. We take `newstest2013` as validation set and `newstest2014` as test set. For Chinese-English task, we use WMT 2017 (Bojar et al., 2014) Chinese-English dataset while take `newsdev2017` as development set and `newstest2017` as test set. The Chinese-English training set contains 24.8 M sentence pairs. We use byte-pair encoding (Sennrich et al., 2015) with 32k merges to encode sentences for all language individually. We evaluate our models by case insensitive BLEU score with the script `multi-bleu.perl` for En-De task and toolkit `sacrebleu` (Post, 2018) for Zh-En task.

Optimization We trained our base model on 4 NVIDIA V100 GPUs with each batch contains about 16384 tokens on each device. For big model, we use 8 NVIDIA V100 GPUs and set the batch size to 8192. The Adam optimizer is applied for optimization. The learning rate is set to 7E-4 and 5E-4 for base and big model respectively. We train the base model for 100k steps and the big model for 200k steps. We use the standard learning rate decay policy proposed by Vaswani et al. (2017). To train additional weights (head weights and FFN weights), we use an extra Adam optimizer with constant learning rate 1E-4 and train for 2k steps. All other parameters are fixed when we train additional weights. Additional weights are not trained in other circumstances. In decoding phase, we set the beam size to be 4 and the length penalty α (Wu et al., 2016) to be 0.6 for En-De task and 1.2 for Zh-En task.

C Experiments Details

C.1 Quick Convergence of Additional Weights

Since we train additional weights to rank the importance of attention heads, an interesting question is how many steps do we need to get a stable pruning decision? As each pruning decision can be noted as a binary mask, with the pruned head marked as 1 and kept one marked as 0. During the process of weight training, inspired by You et al. (2019), we can visualize mask distances between steps to detect whether the weight training converges.

As shown in Figure 6, we find that the mask converge in less than 200 steps of score training, which is negligible compared with the training of model parameters. When we train head weights for the same model several times, the finally converged masks are almost the same, indicating the mask is totally dependent on model parameters and robust to the randomness during training.

C.2 Detailed Main Results and Curves

For fair comparison on total training time, we fix the training steps in total. We first train the Post-Pruning setting with enough steps to convergence, and then prune the model and train the pruned model. The process of training pruned model in Early-Pruning is extended to meet the amount requirement of total steps. We also train the model

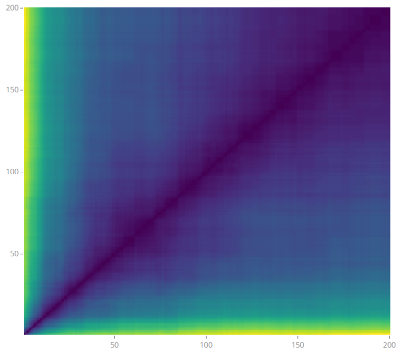

Figure 6: Binary mask distance between steps when training weights at step 30k for base model on En-De dataset. The mask keeps almost unchanged in the last few steps.

of Pre-Pruning with more steps for fair comparison with previous settings. All results are listed in Table 3. The table shows that although the training time of our method is far beyond convergence, it stills costs less time than Post-Pruning under fair comparison. The training curves of En-De base model, Zh-En base model and Zh-En big model are showed in Figure 8, Figure 9 and Figure 10. In all cases our method converges much earlier than Post-Pruning.

C.3 Compare with Pruning Gradually

A natural idea is that we can prune the model more than one times during training. We try this idea and show the results in Table 5 and notated as "Gradually-Pruning". To compare with our previous experiment, we prune the model from step 30k, and prune 2% of all attention heads and corresponding FFN dimensions every 2k steps for 18 times. The final pruned ratio is nearly 30% then. We train additional weights alone for 200 steps before every pruning time, and these weights are reinitialized to 1 before each time we train them. The experiment shows that pruning gradually performs no better than pruning just once, and only behaves more complicated.

C.4 Results of Multiple Runs for Different Pruning Moments

The final results evaluated on BLEU score face turbulence in practice. We run each experiment for 3 times to observe the stable trend via averaged values. We list all results in Table 4.

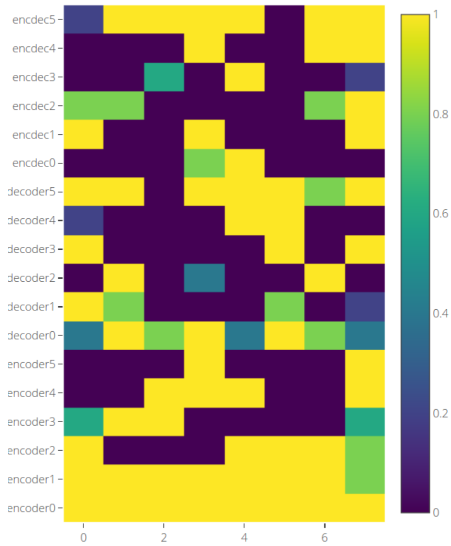

Figure 7: Average pruning decision of 5 independent runs of training weights from same model. Pruning decisions highly focuses on particular heads. Values of most heads in the figure are 0 (never chosen to prune) or 1 (always chosen to prune), indicating the pruning decision of our method is independent of randomness.

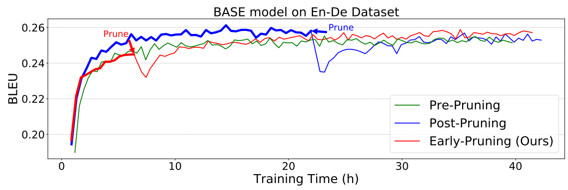

Figure 8: Training curves of base models on En-De dataset.

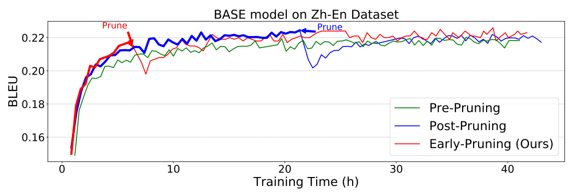

Figure 9: Training curves of base models on Zh-En dataset.

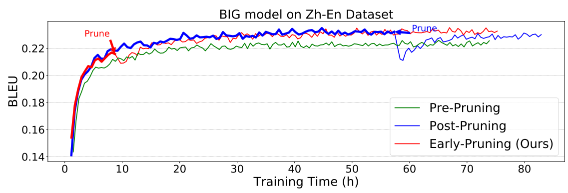

Figure 10: Training curves of big models on Zh-En dataset.

	Steps	En-De Params	BLEU	Time	Zh-En Params	BLEU	Time
BASE model							
Transformer	100k	98.7M	27.54	-	93.2M	23.6	-
Post-Pruning	100k + 100k	84.8M	27.15	42h3m	77.0M	23.3	40h57m
Early-Pruning	30k + 170k	83.8M	27.13	39h59m	78.9M	23.2	39h36m
Pre-Pruning	200k	86.0M	26.66	39h21m	79.9M	22.9	38h30m
BIG model							
Transformer	200k	285.4M	29.01	-	274.5M	23.9	-
Post-Pruning	200k + 100k	223.2M	28.24	83h29m	227.8M	23.6	81h57m
Early-Pruning	30k + 270k	230.0M	28.14	73m29m	222.9M	23.7	74h53m
Pre-Pruning	300k	228.0M	27.60	72h38m	223.0M	23.4	73h30m

Table 3: Results on WMT14 En-De task and WMT17 Zh-En task.

Pruning Step	Prune 30% Run1	Run2	Run3	Avg. BLEU	Prune 50% Run1	Run2	Run3	Avg. BLEU
0k	26.23	27.07	26.69	26.66	25.97	26.13	26.03	26.04
10k	26.88	27.04	26.80	26.91	26.35	26.31	26.21	26.29
20k	27.03	27.00	26.84	26.96	26.20	26.28	26.52	26.29
30k	27.14	27.36	26.88	27.13	26.48	26.43	26.58	26.50
40k	27.22	27.12	27.12	27.15	26.59	26.70	26.69	26.66
50k	27.09	27.04	27.25	27.13	26.67	26.62	26.67	26.65
100k	27.26	27.19	27.10	27.18	26.68	26.70	26.71	26.70

Table 4: All results of 3 runs when change prune step.The pruning ratio is 30% and 50%. The results are evaluated on WMT14 En-De dataset.

	Params	BLEU
Early-Pruning	85.3M	27.13
Gradually-Pruning	91.0M	27.12

Table 5: Compare with prune gradually. We prune the model every 2k steps for 18 times from step 30k with 2% of the parameters each time.