Successfully Applying the Stabilized Lottery Ticket Hypothesis to the Transformer Architecture

Christopher Brix, Parnia Bahar, Hermann Ney

Human Language Technology and Pattern Recognition Group Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
`<surname>@i6.informatik.rwth-aachen.de`

Abstract

Sparse models require less memory for storage and enable a faster inference by reducing the necessary number of FLOPs. This is relevant both for time-critical and on-device computations using neural networks. The stabilized lottery ticket hypothesis states that networks can be pruned after none or few training iterations, using a mask computed based on the unpruned converged model. On the transformer architecture and the WMT 2014 English→German and English→French tasks, we show that stabilized lottery ticket pruning performs similar to magnitude pruning for sparsity levels of up to 85%, and propose a new combination of pruning techniques that outperforms all other techniques for even higher levels of sparsity. Furthermore, we confirm that the parameter's initial sign and not its specific value is the primary factor for successful training, and show that magnitude pruning cannot be used to find winning lottery tickets.

1 Introduction

Current neural networks are heavily growing in depth, with many fully connected layers. As every fully connected layer includes large matrices, models often contain millions of parameters. This is commonly seen as an over-parameterization (Dauphin and Bengio, 2013; Denil et al., 2013). Different techniques have been proposed to decide which weights can be pruned. In structured pruning techniques (Voita et al., 2019), whole neurons or even complete layers are removed from the network. Unstructured pruning only removes individual connections between neurons of succeeding layers, keeping the global network architecture intact. The first technique directly results in smaller model sizes and faster inference, while the second offers more flexibility in the selection of which parameters to prune. Although the reduction in necessary storage space can be realized using sparse matrix representations (Stanimirovi and Tasic, 2009), most popular frameworks currently do not have sufficient support for sparse operations. However, there is active development for possible solutions (Liu et al., 2015; Han et al., 2016; Elsen et al., 2019). This paper compares and improves several unstructured pruning techniques. The main contributions of this paper are to:

- verify that the stabilized lottery ticket hypothesis (Frankle et al., 2019) performs similar to magnitude pruning (Narang et al., 2017) on the transformer architecture (Vaswani et al., 2017) with 60M parameters up to a sparsity of 85%, while magnitude pruning is superior for higher sparsity levels.

- demonstrate significant improvements for high sparsity levels over magnitude pruning by using it in combination with the lottery ticket hypothesis.

- confirm that the signs of the initial parameters are more important than the specific values to which they are reset, even for large networks like the transformer.

- show that magnitude pruning cannot be used to find winning lottery tickets, i.e., the final mask reached using magnitude pruning is no indicator for which initial weights are most important.

2 Related Work

Han et al. (2015) propose the idea of pruning weights with a low magnitude to remove connections that have little impact on the trained model. Narang et al. (2017) incorporate the pruning into the main training phase by slowly pruning parameters during the training, instead of performing one big pruning step at the end. Zhu and Gupta (2018)

provide an implementation for magnitude pruning in networks designed using the tensor2tensor software (Vaswani et al., 2018).

Frankle and Carbin (2018) propose the lottery ticket hypothesis, which states that dense networks contain sparse sub-networks that can be trained to perform as good as the original dense model. They find such sparse sub-networks in small architectures and simple image recognition tasks and show that these sub-networks might train faster and even outperform the original network. For larger models, Frankle et al. (2019) propose to search for the sparse sub-network not directly after the initialization phase, but after only a few training iterations. Using this adapted setup, they are able to successfully prune networks having up to 20M parameters. They also relax the requirement for lottery tickets so that they only have to beat randomly initialized models with the same sparsity level.

Zhou et al. (2019) show that the signs of the weights in the initial model are more important than their specific values. Once the least important weights are pruned, they set all remaining parameters to fixed values, while keeping their original sign intact. They show that as long as the original sign remains the same, the sparse model can still train more successfully than one with a random sign assignment. Frankle et al. (2020) reach contradicting results for larger architectures, showing that random initialization with original signs hurts the performance.

Gale et al. (2019) compare different pruning techniques on challenging image recognition and machine translation tasks and show that magnitude pruning achieves the best sparsity-accuracy trade-off while being easy to implement.

In concurrent work, Yu et al. (2020) test the stabilized lottery ticket on the transformer architecture and the WMT 2014 English→German task, as well as other architectures and fields.

This paper extends the related works by demonstrating and comparing the applicability of different pruning techniques on a deep architecture for two translation tasks, as well as proposing a new combination of pruning techniques for improved performance.

3 Pruning Techniques

In this section, we give a brief formal definition of each pruning technique. For a more detailed description, refer to the respective original papers.

In the given formulas, a network is assumed to be specified by its parameters θ. When training the network for T iterations, θ_t for $t \in [0, T]$ represents the parameters at timestep t.

Magnitude Pruning (MP) relies on the magnitude of parameters to decide which weights can be pruned from the network. Different techniques to select which parameters are selected for pruning have been proposed (Collins and Kohli, 2014; Han et al., 2015; Guo et al., 2016; Zhu and Gupta, 2018). In this work, we rely on the implementation from Zhu and Gupta (2018) where the parameters of each layer are sorted by magnitude, and during training, an increasing percentage of the weights are pruned. It is important to highlight that MP is the only pruning technique not requiring multiple training runs.

Lottery Ticket (LT) pruning assumes that for a given mask m, the initial network θ_0 already contains a sparse sub-network $\theta_0 \odot m$ that can be trained to the same accuracy as θ_0. To determine m, the parameters of each layer in the converged model θ_T are sorted by magnitude, and m is chosen to mask the smallest ones such that the target sparsity s_T is reached. We highlight that even though m is determined using θ_T, it is then applied to θ_0 before the sparse network is trained. To reach high sparsity without a big loss on accuracy, Frankle and Carbin (2018) recommend to prune iteratively, by training and resetting multiple times.

Stabilized Lottery Ticket (SLT) pruning is an adaptation of LT pruning for larger models. Frankle et al. (2019) propose to apply the computed mask m not to the initial model θ_0, but to an intermediate checkpoint θ_t where $0 < t \ll T$ is chosen to be early during the training. They recommend to use $0.001T \leq t \leq 0.07T$ and refer to it as iterative magnitude pruning with rewinding. We highlight that Frankle et al. (2019) always choose θ_t from the first, dense model, while this work choses θ_t from the last pruning iteration.

Constant Lottery Ticket (CLT) pruning assumes that the specific random initialization is not important. Instead, only the corresponding choice of signs affects successful training. To show this, Zhou et al. (2019) propose to compute $\theta_t \odot m$ as in SLT pruning, but then to train $f(\theta_t \odot m)$ as the sparse model. Here, f sets all remaining parameters p in each layer l to $\text{sign}(p) \cdot \alpha_l$, i.e., all param-

eters in each layer have the same absolute value, but their original sign. In all of our experiments, α_l is chosen to be $\alpha_l = \sqrt{\frac{6}{n_{l_{in}} + n_{l_{out}}}}$ where $n_{l_{in}}$ and $n_{l_{out}}$ are the respective incoming and outgoing connections to other layers.

SLT-MP is a new pruning technique, proposed in this work. It combines both SLT pruning and MP in the following way: First, SLT pruning is used to find a mask m with intermediate sparsity s_i. This might be done iteratively. $\theta_t \odot m$ with sparsity s_i is then used as the initial model for MP (i.e., $\theta'_0 = \theta_t \odot m$). Here, in the formula for MP, $s_0 = s_i$. We argue that this combination is beneficial, because in the first phase, SLT pruning removes the most unneeded parameters, and in the second phase, MP can then slowly adapt the model to a higher sparsity.

MP-SLT is analogue to SLT-MP: First, MP is applied to compute a trained sparse network θ_T with sparsity s_i. This trained network directly provides the corresponding mask m. $\theta_t \odot m$ is then used for SLT pruning until the target sparsity is reached. This pruning technique tests whether MP can be used to find winning lottery tickets.

4 Experiments

We train the models on the WMT 2014 English→German and English→French datasets, consisting of about 4.5M and 36M sentence pairs, respectively. `newstest2013` and `2014` are chosen to be the development and test sets.

All experiments have been performed using the base transformer architecture as described in (Vaswani et al., 2017).[1] The models are trained for 500k iterations on a single v3-8 TPU, saving checkpoints every 25k iterations. For all experiments, we select the best model based on the BLEU score on the development set. For MP, we only evaluate the last 4 checkpoints, as earlier checkpoints do not have the targeted sparsity. Intermediate MP sparsity levels s_t are computed as $s_t = s_T + \min\{0, (s_0 - s_T)(1 - \frac{t}{400000})^3\}$ (Zhu and Gupta, 2018). For efficiency reasons, weights are only pruned every 10k iterations. Unless stated otherwise, we start with initial sparsity $s_0 = 0$. The final sparsity s_T is individually given for each experiment.

---

[1] Using the hyperparameters in `transformer_base_v3` in https://github.com/tensorflow/tensor2tensor/blob/838f1a99e24a9391a8faf6603e90d476444110a0/tensor2tensor/models/transformer.py with the corresponding adaptations for TPUs.

We prune only the matrices, not biases. We report the approximate memory consumption of all trained models using the Compressed Sparse Column (CSC) format (Stanimirovi and Tasic, 2009), which is the default for sparse data storage in the SciPy toolkit (Virtanen et al., 2020).

Our initial experiments have shown that Adafactor leads to an improvement of 0.5 BLEU compared to Adam. Hence, we select it as our optimizer with a learning rate of $lr(t) = \frac{1}{\max(t,w)}$ for $w = 10k$ warmup steps. We note that this differs from the implementation by Gale et al. (2019), in which Adam has been used. We highlight that for all experiments that require a reset of parameter values (i.e., LT, SLT, CLT, SLT-MP, and MP-SLT), we reset t to 0, to include the warmup phase in every training run.

A shared vocabulary of 33k tokens based on word-pieces (Wu et al., 2016) is used. The reported case-sensitive, tokenized BLEU scores are computed using SacreBLEU (Post, 2018), TER scores are computed using MultEval (Clark et al., 2011). All results are averaged over two separate training runs. For all experiments that require models to be reset to an early point during training, we select a checkpoint after 25k iterations.

All iterative pruning techniques except SLT-MP are pruned in increments of 10 percentage points up to 80%, then switching to 5 points increments, and finally pruning to 98% sparsity. SLT-MP is directly trained using SLT pruning to 50% and further reduced by SLT to 60%, before switching to MP.

5 Experimental Results

In this section, we evaluate the experimental results for English→German and English→French translation given in Tables 1 and 2 to provide a comparison between the different pruning techniques described in Section 3.

MP Tables 1 and 2 clearly show a trade-off between accuracy and network performance. For every increase in sparsity, the performance degrades accordingly. We especially note that even for a sparsity of 50%, the baseline performance cannot be achieved. In contrast to all other techniques in this paper, MP does not require any reset of parameter values. Therefore, the training duration is not increased.

LT Frankle and Carbin (2018) test the LT hypothesis on the small ResNet-50 architecture (He et al., 2016) which is applied to ImageNet (Russakovsky

Table 1

Sparsity	Memory	MP		LT		SLT		CLT		SLT-MP		MP-SLT	
		BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
0%	234 MB	26.8	64.5	26.8	64.5	26.8	64.5	26.8	64.5	26.8	64.5	26.8	64.5
10%	226 MB	26.8	**64.5**	26.7	64.6	26.8	64.9	**26.9**	64.7	n/a	n/a	26.8	64.5
20%	206 MB	26.7	**64.5**	26.2	65.3	26.9	64.6	**27.0**	**64.5**	n/a	n/a	26.7	64.5
30%	184 MB	26.4	65.0	26.0	65.3	**26.9**	64.8	**26.9**	64.7	n/a	n/a	26.4	65.0
40%	161 MB	26.5	**64.8**	25.8	65.7	**27.1**	65.1	26.8	65.0	n/a	n/a	26.5	64.8
50%	137 MB	26.4	65.0	25.4	66.3	26.6	65.2	**26.7**	65.2	26.4†	64.9†	26.4	65.0
60%	112 MB	25.9	65.5	24.9	66.5	26.4	65.7	**26.8**	65.0	26.4†	65.1†	25.9	65.5
70%	86 MB	25.7	65.8	24.2	67.6	25.6	66.9	**26.2**	65.8	26.2‡	65.3‡	25.6	66.0
80%	59 MB	24.8	66.8	23.2	68.4	24.8	67.7	24.1	67.9	**25.6**‡	**65.9**‡	24.6	67.2
85%	46 MB	23.9	67.7	22.3	69.8	23.7	68.5	23.7	68.0	**24.9**‡	**66.4**‡	23.9	67.9
90%	31 MB	22.9	69.0	20.9	72.0	21.7	71.4	21.6	70.6	**23.5**‡	**68.4**‡	22.4	69.8
95%	17 MB	20.2	72.9	18.1	75.4	17.4	77.1	18.2	73.3	**20.5**‡	**72.3**‡	18.5	75.5
98%	7 MB	15.8	78.9	13.3	81.2	11.0	86.9	14.6	**78.2**	**16.1**‡	79.2‡	13.5	82.6

Table 1: En→De translation: BLEU[%] and TER[%] scores of the final model at different sparsity levels, evaluated on `newstest2014`. For SLT-MP, models marked with † are trained with SLT pruning, models marked with ‡ are trained with MP. For MP-SLT, the MP model with 60% sparsity was used for SLT pruning. For each sparsity level, the best score is highlighted.

Table 2

Sp.	MP		SLT		CLT		SLT-MP	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
0%	39.3	57.2	39.3	57.2	39.3	57.2	39.3	57.2
10%	39.3	**57.2**	39.3	57.4	**39.4**	57.4	n/a	n/a
20%	**39.3**	57.2	**39.3**	**57.1**	**39.3**	57.2	n/a	n/a
30%	39.3	57.1	**39.8**	**56.7**	39.7	56.9	n/a	n/a
40%	38.8	57.8	**39.7**	**56.9**	39.2	57.3	n/a	n/a
50%	38.8	57.7	39.2	57.4	**39.4**	57.4	39.0†	**57.3**†
60%	38.5	57.9	39.0	57.6	**39.2**	57.5	**39.2**†	57.4†
70%	38.2	58.4	38.4	58.3	**38.9**	**57.8**	38.5‡	58.2‡
80%	37.5	59.1	37.4	59.3	37.3	59.2	**38.0**‡	**58.7**‡
85%	37.0	**59.6**	36.9	**59.6**	35.7	61.1	**37.4**‡	**59.6**‡
90%	35.6	61.4	34.7	62.1	33.7	62.9	**35.9**‡	**60.4**‡
95%	32.7	63.8	28.5	68.0	29.6	65.7	**33.1**‡	**63.1**‡
98%	27.1	69.6	21.8*	73.9*	19.6	75.9	**27.3**‡	**68.9**‡

Table 2: En→Fr translation: BLEU[%] and TER[%] scores of the final model at different sparsity levels, evaluated on `newstest2014`. For SLT-MP, models marked with † are trained with SLT pruning, models marked with ‡ are trained with MP. (∗) indicates a result of a single run, as the second experiment failed. For each sparsity level, the best score is highlighted.

et al., 2015). Gale et al. (2019) apply LT pruning to the larger transformer architecture and the translation task WMT 2014 English→German, noting that it has been outperformed by MP. As seen in Table 1, simple LT pruning is outperformed by MP at all sparsity levels. Because LT pruning is an iterative process, training a network with sparsity 98% requires to train and reset the model 13 times, causing a big training overhead without any gain in performance. Therefore, simple LT pruning cannot be recommended for complex architectures.

SLT The authors of the SLT hypothesis (Frankle et al., 2019) state that after 0.1-7% of the training, the intermediate model can be pruned to a sparsity of 50-99% without serious impact on the accuracy. As listed in Tables 1 and 2, this allows the network to be pruned up to 60% sparsity without a significant drop in BLEU, and is on par with MP up to 85% sparsity.

As described in Section 4, for resetting the models, a checkpoint after $t = 25k$ iterations is used. For a total training duration of 500k iterations, this amounts to 5% of the training and is therefore within the 0.1-7% bracket given by Frankle et al. (2019). For individual experiments, we have also tried $t \in \{12.5k, 37.5k, 500k\}$ and have gotten similar results to those listed in this paper. It should be noted that for the case $t = 500k$, SLT pruning becomes a form of MP, as no reset happens anymore. We propose a more thorough hyperparameter search for the optimal t value as future work.

Importantly, we note that the magnitude of the parameters in both the initial and the final models increases with every pruning step. This causes the model with 98% sparsity to have weights greater than 100, making it unsuitable for checkpoint averaging, as the weights become too sensitive to minor changes. Yu et al. (2020) report that they do successfully apply checkpoint averaging. This might

be because they choose θ_t from the dense training run for resetting, while we choose θ_t from the most recent sparse training.

CLT The underlying idea of the LT hypothesis is, that the untrained network already contains a sparse sub-network which can be trained individually. Zhou et al. (2019) show that only the signs of the remaining parameters are important, not their specific random value. While Zhou et al. (2019) perform their experiments on MNIST and CIFAR-10, we test this hypothesis on the WMT 2014 English→German translation task using a deep transformer architecture.

Surprisingly, CLT pruning outperforms SLT pruning on most sparsity levels (see Table 1). By shuffling or re-initializing the remaining parameters, Frankle and Carbin (2018) have already shown that LT pruning does not just learn a sparse topology, but that the actual parameter values are of importance. As the good performance of the CLT experiments indicates that changing the parameter values is of little impact as long as the sign is kept the same, we verify that keeping the original signs is indeed necessary. To this end, we randomly assign signs to the parameters after pruning to 50% sparsity. After training, this model scores 24.6% BLEU and 67.5% TER, a clear performance degradation from the 26.7% BLEU and 65.2% TER given in Table 1. Notably, this differs from the results by Frankle et al. (2020), as their results indicate that the signs alone are not enough to guarantee good performance.

SLT-MP Across all sparsity levels, the combination of SLT pruning and MP outperforms all other pruning techniques. For high sparsity values, SLT-MP models are also superior to the SLT models by Yu et al. (2020), even though they start off from a better performing baseline. We hypothesize that by first discarding 60% of all parameters using SLT pruning, MP is able to fine-tune the model more easily, because the least useful parameters are already removed.

We note that the high weight magnitude for sparse SLT models prevents successful MP training. Therefore, we have to reduce the number of SLT pruning steps by directly pruning to 50% in the first pruning iteration. However, as seen by comparing the scores for 50% and 60% sparsity on SLT and SLT-MP, this does not hurt the SLT performance.

For future work, we suggest trying different sparsity values s_i for the switch between SLT and MP.

MP-SLT Switching from MP to SLT pruning causes the models to perform worse than for pure MP or SLT pruning. This indicates that MP cannot be used to find winning lottery tickets.

6 Conclusion

In conclusion, we have shown that the stabilized lottery ticket (SLT) hypothesis performs similar to magnitude pruning (MP) on the complex transformer architecture up to a sparsity of about 85%. Especially for very high sparsities of 90% or more, MP has proven to perform reasonably well while being easy to implement and having no additional training overhead. We also have successfully verified that even for the transformer architecture, only the signs of the parameters are important when applying the SLT pruning technique. The specific initial parameter values do not significantly influence the training. By combining both SLT pruning and MP, we can improve the sparsity-accuracy trade-off. In SLT-MP, SLT pruning first discards 60% of all parameters, so MP can focus on fine-tuning the model for maximum accuracy. Finally, we show that MP cannot be used to determine winning lottery tickets.

In future work, we suggest performing a hyperparameter search over possible values for t in SLT pruning (i.e., the number of training steps that are not discarded during model reset), and over s_i for the switch from SLT to MP in SLT-MP. We also recommend looking into why CLT pruning works in our setup, while Frankle et al. (2020) present opposing results.

References

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Maxwell D. Collins and Pushmeet Kohli. 2014. Memory bounded deep convolutional networks. *CoRR*, abs/1412.1442.

Yann N. Dauphin and Yoshua Bengio. 2013. Big neural networks waste capacity. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Misha Denil, Babak Shakibi, Laurent Dinh, MarcAurelio Ranzato, and Nando de Freitas. 2013. Predicting parameters in deep learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS13, page 21482156, Red Hook, NY, USA. Curran Associates Inc.

Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. 2019. Fast sparse convnets.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2019. The lottery ticket hypothesis at scale. *CoRR*, abs/1903.01611.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. 2020. The early phase of neural network training. In *International Conference on Learning Representations*.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574.

Yiwen Guo, Anbang Yao, and Yurong Chen. 2016. Dynamic network surgery for efficient dnns. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1379–1387. Curran Associates, Inc.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: efficient inference engine on compressed deep neural network. *CoRR*, abs/1602.01528.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. 2015. Sparse convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sharan Narang, Greg Diamos, Shubho Sengupta, and Erich Elsen. 2017. Exploring sparsity in recurrent neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Ivan Stanimirovi and Milan Tasic. 2009. Performance comparison of storage formats for sparse matrices. *Facta Universitatis. Series Mathematics and Informatics*, 24.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199, Boston, MA. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren

Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

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, Lukasz 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. *CoRR*, abs/1609.08144.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2020. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. In *International Conference on Learning Representations*.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3597–3607. Curran Associates, Inc.

Michael Zhu and Suyog Gupta. 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net.