What's Hidden in a One-layer Randomly Weighted Transformer?

Sheng Shen[†*], Zhewei Yao[†*], Douwe Kiela[‡], Kurt Keutzer[†] and Michael W. Mahoney[†]
[†]UC Berkeley; [‡]Facebook AI Research
{sheng.s,zheweiy}@berkeley.edu, dkiela@fb.com

Abstract

We demonstrate that, hidden within one-layer randomly weighted neural networks, there exist subnetworks that can achieve impressive performance, without ever modifying the weight initializations, on machine translation tasks. To find subnetworks for one-layer randomly weighted neural networks, we apply different binary masks to the same weight matrix to generate different layers. Hidden within a one-layer randomly weighted Transformer, we find that subnetworks that can achieve 29.45/17.29 BLEU on IWSLT14/WMT14. Using a fixed pre-trained embedding layer, the previously found subnetworks are smaller than, but can match 98%/92% (34.14/25.24 BLEU) of the performance of, a trained Transformer$_{\text{small/base}}$ on IWSLT14/WMT14. Furthermore, we demonstrate the effectiveness of larger and deeper transformers in this setting, as well as the impact of different initialization methods.[1]

1 Introduction

Modern deep learning often trains millions or even billions of parameters (Devlin et al., 2018; Shoeybi et al., 2019; Raffel et al., 2019; Brown et al., 2020) to deliver good performance for a model. Recently, Frankle and Carbin (2018); Frankle et al. (2020) demonstrated that these over-parameterized networks contain sparse subnetworks, when trained in isolation, that can achieve similar or better performance than the original model.

Furthermore, recent studies revisit the initialization stage of finding these subnetworks in vision models (Zhou et al., 2019; Ramanujan et al., 2020). Such a mask, which is used to mask out a part of the entire network to those subnetworks, is referred to as a "Supermask." That is to say, subnetworks of a *randomly weighted neural network* (NN) can achieve competitive performance, which may act as a good "prior" (Gaier and Ha, 2019) and connect to the long history of leveraging random features (Gamba et al., 1961; Baum, 1988) and/or random kernel methods (Rahimi and Recht, 2008, 2009) in machine learning. Here, we examine the following question: how does a fully randomized natural language processing (NLP) model perform in the multi-layer setting, and particularly in the (so far under-explored) one-layer setting?

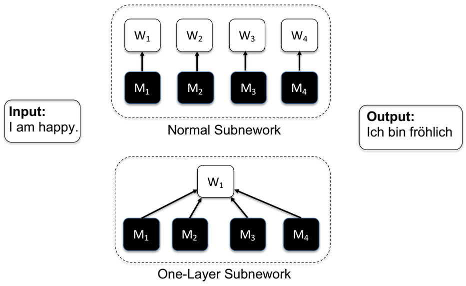

Figure 1: Illustration plot for a normal subnetwork and a one-layer subnetwork.

In this work, we first validate that there exist subnetworks of standard randomly weighted Transformers (Reservoir Transformers in (Shen et al., 2021)) that can perform competitively with fully-weighted alternatives on machine translation and natural language understanding tasks. With 50% randomized weights remaining, we found a subnetwork that can reach 29.45/17.29 BLEU on IWSLT14/WMT14, respectively. We also investigate the special case of finding subnetworks in one-layer randomly weighted Transformers (see Fig. 1). To obtain the subnetworks, we repeatedly apply the same randomized Transformer layer several times with different Supermasks. The resulting subnetwork of a one-layer randomly-weighted Transformer has similar performance as the multi-layer counterparts with a 30% lower memory footprint. We also study the

[1]We released the source code at https://github.com/sIncerass/one_layer_lottery_ticket.
*Equal contribution.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2914–2921
November 7–11, 2021.

impact of different depths/widths of Transformers along with the effectiveness of two initialization methods. Finally, using the pre-trained embedding layers, we find that the subnetworks hidden in one layer randomly weighted Transformer$_{\text{wide/wider}}$ are smaller than, but can match 98%/92% of the performance of, a trained Transformer$_{\text{small/base}}$ on IWSLT14/WMT14. We hope our findings can offer new insights for understanding Transformers.

2 Related Work

Lottery Tickets Hypothesis. Frankle and Carbin (2018) found that NNs for computer vision contain subnetworks that can be effectively trained from scratch when reset to their initialization. Subsequent works (Zhou et al., 2019; Ramanujan et al., 2020; Wortsman et al., 2020) demonstrated that so-called winning tickets can achieve performance without training, where the mask for finding the subnetwork at initialization is called "supermask." In NLP, previous works find that matching subnetworks exist early in training with Transformers (Yu et al., 2019), LSTMs (Renda et al., 2020), and fully-weighted per-trained BERT (Chen et al., 2020; Prasanna et al., 2020) or Vison-and-Language model (Gan et al., 2021), but not at *initialization*.

Random Feature. In the early days of neural networks, fixed random layers (Baum, 1988; Schmidt et al., 1992; Pao et al., 1994) have been studied in reservoir computing (Maass et al., 2002; Jaeger, 2003; Lukoševičius and Jaeger, 2009), "random kitchen sink" kernel machines (Rahimi and Recht, 2008, 2009), and so on. Recently, random features have also been extensively explored for modern neural networks in deep reservoir computing networks (Scardapane and Wang, 2017; Gallicchio and Micheli, 2017; Shen et al., 2021), random kernel feature (Peng et al., 2021; Choromanski et al., 2020), and applications in text classification (Conneau et al., 2017; Wieting and Kiela, 2019), summarization (Pilault et al., 2020) and probing (Voita and Titov, 2020).

Compressing Transformer. A wide range of neural network compression techniques have been applied to Transformers. This includes pruning (Fan et al., 2019; Michel et al., 2019; Sanh et al., 2020; Yao et al., 2021) where parts of the model weights are dropped, parameter-sharing (Lan et al., 2020; Dehghani et al., 2018; Bai et al., 2019) where the same parameters are used in different parts of a model, quantization (Shen et al., 2020; Li et al., 2020) where the weights of the Transformer model are represented with fewer bits, and distilliation (Sun et al., 2020; Jiao et al., 2020) where a compact student model is trained to mimic a larger teacher model. To find the proposed subnetwork at initialization, we develop our method in the spirit of parameter sharing and pruning.

3 Methodology

Finding a Supermask for Randomly Weighted Transformer. In a general pruning framework, denote weight matrix as $\mathbf{W} \in \mathbb{R}^{d\times d}$ ($\mathbf{W}$ could be a non-square matrix), input as $x \in \mathbb{R}^d$ and the network as $f(x; \mathbf{W})$. A subnetwork defined is $f(x; \mathbf{W} \odot \mathbf{M})$, where $\mathbf{M} \in \mathbb{R}^{d\times d}$ is a binary matrix and $\odot$ is the element-wise product. To find the subnetwork for a randomly weighted network, $\mathbf{M} \in \mathbb{R}^{d\times d}$ is trained while $\mathbf{W}$ is kept at a random initialization. Following Ramanujan et al. (2020), denote $\mathbf{S} \in \mathbb{R}^{d\times d}$ as the associated importance score matrix of $\mathbf{W}$, which is learnable during training. We keep top-k percents of weights by the importance score of $\mathbf{S}$ to compute $\mathbf{M}$, i.e.,

$$\mathbf{M} = \text{Top}_k(\mathbf{S}), \text{where Top}_k(\mathbf{S}_{i,j}) = \begin{cases} 1 & \mathbf{S}_{i,j} \text{ in top k\%,} \\ 0 & \text{else.} \end{cases}$$

Note that Top_k is an undifferentiated function. To enable training of $\mathbf{S}$, we use the straight-through gradient estimator (Bengio et al., 2013), in which Top_k is treated as the identity in backpropagation. During inference, we can simply construct and store the binary Supermask $\mathbf{M}$ and the floating-point $\mathbf{W}$ while dropping $\mathbf{S}$ for future usage.

One-layer randomly weighted Transformer. We use the Transformer architecture (see Vaswani et al. (2017) for more details). For a general randomly weighted Transformer model with Supermask, there exist $\mathbf{M}_l$s and $\mathbf{W}_l$s for all layers $l \in \{1, ...L\}$. Due to the natural property of layer stacking in Transformers, all $\mathbf{W}_l$s have the same shape with the same initialization method. This leads to an unexplored question: "What's hidden in a one-layer (instead of L-layer) randomly weighted transformer?"

Let us use a toy example to explain why there is no need for L redundant $\mathbf{W}_l$s. Assume that, for a random weighted matrix $\mathbf{W}_l$, the probability that it has a "good" subnetwork is p[2]. Furthermore, assume that for two different layers, the probability

[2]Here, the "good" can be any defined metric, e.g., $(\mathbf{M} \odot \mathbf{W}_l)$x $\approx \mathbf{W}^*$x for all x and a pre-defined $\mathbf{W}^*$.

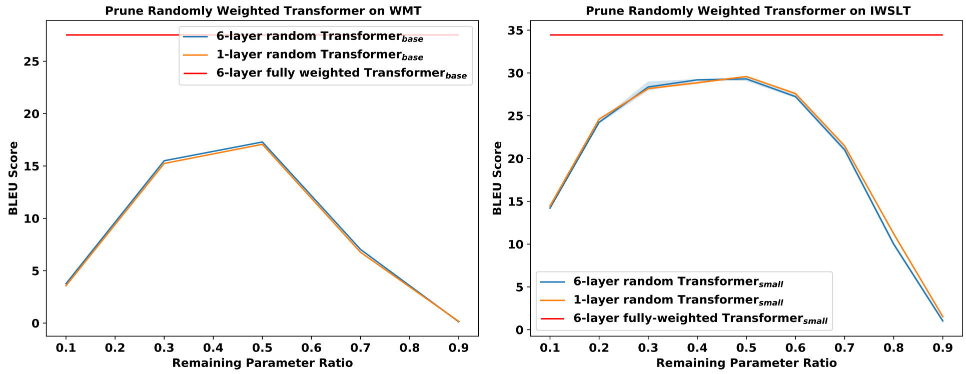

Figure 2: Prune Randomly Weighted Transformer performance on WMT14 (left) and IWSLT14 (right).

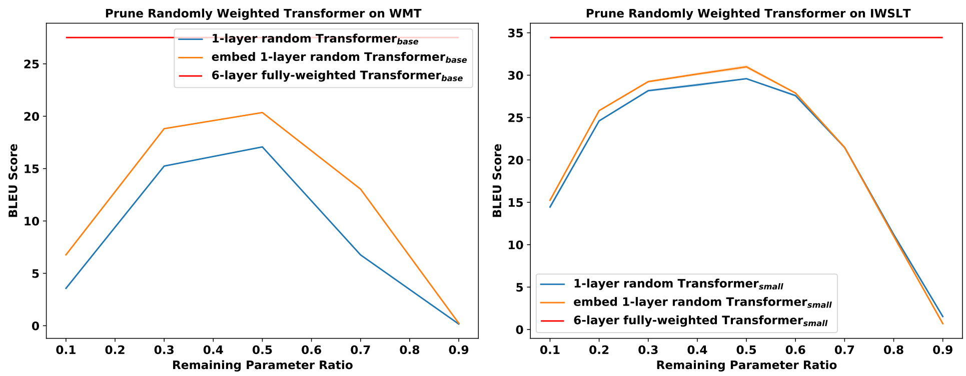

Figure 3: The effectiveness of pre-trained embedding layers on WMT14 (left) and IWSLT14 (right).

that both have the "good" subnetworks is independent. Then for L different layers, the probability that all $\mathbf{W}_l$s have the "good" subnetworks is p^L. Meanwhile, since $\mathbf{W}_1$ has the same initialization method as $\mathbf{W}_l$, the probability that $\mathbf{W}_1$ has a "good" subnetwork for l-th layer is also p. Thus, for L different layers, the probability that using $\mathbf{W}_1$ to generate all "good" subnetworks is also p^L.

In this paper, we investigate the scenario where one randomized layer is applied for L times repeatedly with L different Supermasks. As a result, this can reduce the memory footprint since all Supermasks can be stored in the binary format.

4 Experiments

Model Architecture. For model architectures, we experiment with Transformer$_{\text{small}}$ and Transformer$_{\text{base}}$, following the same setting as in Ott et al. (2018): 6 encoder layers and 6 decoder layers on IWSLT14 and WMT14. We also vary the depth and width of the Transformer model on machine translation tasks. On IWSLT14, we use 3 different random seeds and plot the mean accuracy $\pm$ one standard deviation. All the embedding layers (including the final output projection layer) are also randomized and pruned unless otherwise specified. Moreover, on all figures, the "fully-weighted model" denotes the standard full model (all weights remaining).

Machine Translation results. In Fig. 2, we present results for directly pruning a randomly weighted Transformer on IWSLT14 and WMT14 tasks. Specifically, we vary the ratio of remaining parameters in the randomized model.

As can be seen, there is no significant performance difference between a one-layer random Transformer versus a 6-layer standard random Transformer across different percents of remaining weights on IWSLT14 and WMT14. We also observe that having the remaining randomized weight percents approach 0 or 100 leads to the worst performance across the settings. This is expected since the outputs will be random when we have 100% randomized weights, and the model will not perform well when only limited weights are unpruned (close to 0%). The best performing subnetwork of a one-layer randomized Transformer has 50% weights remained. Connected to the search space of the employed method where we are choosing σ% out of 100% randomized weights, $\sigma = 50$ leads to the largest search space.

Effectiveness of Pre-trained Embeddding layers. Embedding layers are critical since they can be viewed as the inputs for an NLP model,

Task	Model	BLEU	Memory	Remaining Param Ratio	Param (no mask)
IWSLT	$\text{Trans}_{\text{small}}$	34.66 ($\pm$0.11)	148MB	100.0	39M
	One-layer Random $\text{Trans}_{\text{small}}$	30.95 ($\pm$0.12)	28MB	50.0	7M
	One-layer $\text{Trans}_{\text{wide}}$	34.14 ($\pm$0.08)	71MB	50.0	18M
	One-layer Random $\text{Trans}_{\text{deep}}$	31.51 ($\pm$0.10)	29MB	50.0	7M
WMT	Trans-base	27.51	328MB	100.0	86M
	One-layer Random $\text{Trans}_{\text{base}}$	20.35	96MB	50.0	25M
	One-layer Random $\text{Trans}_{\text{wider}}$	25.24	227MB	50.0	57M
	One-layer Random $\text{Trans}_{\text{deeper}}$	21.76	98MB	50.0	25M

Table 1: Machine Translation result for a fully-weighted Transformer versus one-layer random Transformer with pre-trained embedding layer (retain 50% weights). IWSLT14 results are averaged over 3 random seeds, standard deviations are in brackets.

which are analogous to the image pixels in vision. Plenty of prior studies have explored how to obtain the pre-trained embedding in an unsupervised way (Mikolov et al., 2013; Pennington et al., 2014). We experiment with this practical setting where we could have access to the encoder/decoder embedding layers, which are pre-trained from the public checkpoint in fairseq[3], and we present the results in Fig. 3. We observe a significant performance boost for a one-layer randomized transformer across different remaining weights. The difference is much larger for the bigger WMT14 dataset (around +3.0 BLEU for WMT14 and +1.0 BLEU for IWSLT14). The best one-layer randomized Transformer reaches 89%/74% of the fully-weighted Transformer performance on IWSLT14/WMT14, respectively.

Effectiveness of Depth and Width. In Tab. 1, we report the parameter size, BLEU score, and memory size of different one-layer randomized Transformers with 50% remaining weights, where $\text{Trans}_{\text{deep/deeper}}$ are 12 encoder/decoder layers variant of $\text{Trans}_{\text{small/base}}$. $\text{Trans}_{\text{wide/wider}}$ have 2x hidden size as the $\text{Trans}_{\text{small/base}}$. The results are gathered with pre-trained encoder/decoder embedding layers.[4]

Either increasing the depth or enlarging the width can improve the performance of our one-layer random transformer. Particularly,

[3]https://github.com/pytorch/fairseq/

[4]We use the checkpoint from FairSeq for $\text{Trans}_{\text{base/big}}$ on WMT14, and $\text{Trans}_{\text{small}}$ on IWSLT14 to obtain the pre-trained embedding layer for one-layer $\text{Trans}_{\text{base/wider}}$ and one-layer $\text{Trans}_{\text{small}}$. For one-layer $\text{Trans}_{\text{wide}}$ on IWSLT14, we pre-train fully-weighted model and then dump the embedding layer. $\text{Trans}_{\text{deep/deeper}}$ share the same embedding of the $\text{Trans}_{\text{small/base}}$.

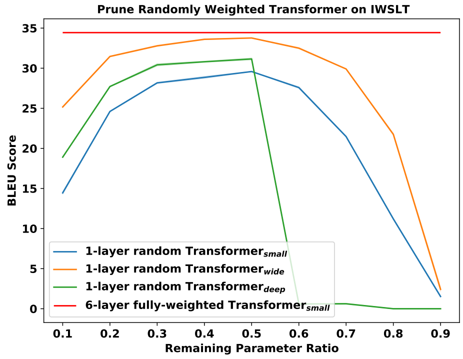

Figure 4: The effectiveness of depth and width.

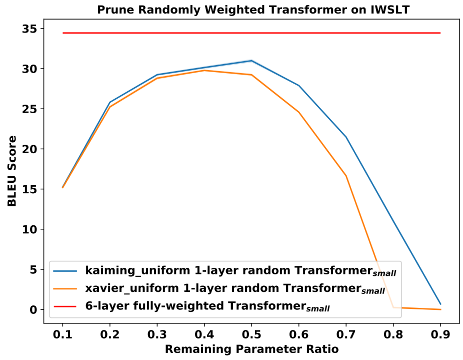

Figure 5: The effectiveness of different initialization.

the deeper transformer can already achieve 79%/90% of the fully-weighted baseline models on WMT14/IWSLT14, respectively. For wider models, those numbers even increase to 92%/98%. This is mainly due to the larger search space introduced by the larger weight matrix. Another important point is that even when we increase/enlarge the depth/width of the model, the total memory consumption of these models is actually smaller than the standard baseline, since we only have one repeated layer and all the masks can be stored in a 1-bit setting.

Furthermore, we explore the effect of the different ratios of remaining parameters for different models on IWSLT14 in Fig. 4. As can be seen, for the wider model, its performance is always better than the standard one across all different settings. However, for the deeper model, there is a sharp transition that happens at 50%–60% remaining parameters. The reason is that, given that our deeper model is twice as deep as the original, when we retain more random parameters (>50%), the probability that the layer has a good "subnetwork" decreases significantly. This will lead the final probability to be p^{2L}_{smaller} ($p_{\text{smaller}} < p$), which is much smaller than p^L (see Section 3).

Different Initialization. Weight initialization is one of the critical components to the success of

Figure 6: Prune Randomly Weighted Transformer performance on QQP .

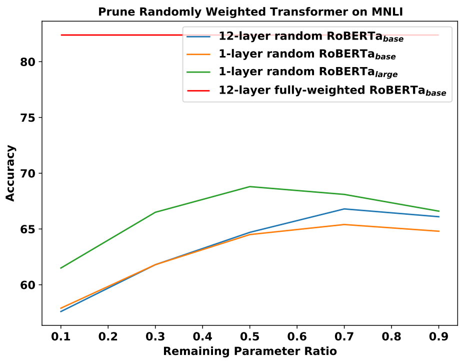

Figure 7: Prune Randomly Weighted Transformer performance on MNLI.

the random feature (Wieting and Kiela, 2019; Ramanujan et al., 2020; Shen et al., 2021). We experiment with kaiming uniform (Ramanujan et al., 2020) and Xavier uniform (Vaswani et al., 2017) initialization methods, and we scale the standard deviation by $\sqrt{1/\sigma}$ when we retain σ randomized weights. As shown in Fig. 5, the performance of the one-layer randomized Transformer decreases when we switch to the Xavier uniform. The degradation becomes larger when more randomized weights retain in the network.

QQP and MNLI results. On QQP and MNLI, we experiment with RoBERTa$_{\text{small}}$ and RoBERTa$_{\text{large}}$, following Liu et al. (2019). We use the pre-trained embedding layer of RoBERTa$_{\text{base/large}}$ (Liu et al., 2019). In Fig. 6 and 7, we show consistent results on QQP and MNLI, except that the best performing one-layer randomly weighted RoBERTa is achieved when we retain 70% randomized weights, it reaches 79%/91% fully-weighted RoBERTa$_{\text{base}}$ accuracy on QQP and MNLI, respectively. The performance approaches 84%/92% of the afore-mentioned fully-weighted model performance when using the larger hidden size with one-layer randomly weighted RoBERTa$_{\text{large}}$.

Implementation Details. We evaluate on IWSLT14 de-en (Cettolo et al., 2015) and WMT14 en-de (Bojar et al., 2014) for machine translation; QQP (Iyer et al., 2017) and MultiNLI-matched (MNLI) (Williams et al., 2017) for natural language understanding.[5] We use 8 Volta V100 GPUs for WMT, and one V100 for IWSLT, QQP, and MNLI. The hyperparameters on IWSLT14 and WMT14 for training a one-layer randomized Transformer were set the same to the best-performing values from Ott et al. (2018) for training fully-weighted Transformer. The QQP and MNLI experiments followed Liu et al. (2019).

5 Conclusions

In this paper, we validate the existence of effective subnetworks in a one-layer randomly weighted Transformer on translation tasks. Hidden within a one-layer randomly weighted Transformer$_{\text{wide/wider}}$ with fixed pre-trained embedding layers, we find there exist subnetworks that are smaller than, but can competitively match, the performance of a trained Transformer$_{\text{small/base}}$ on IWSLT14/WMT14.

Acknowledgements

We thank anonymous reviewers for their comments and suggestions. SS and KK were supported by grants from Samsung, Facebook, and the Berkeley Deep Drive Consortium. We would like to acknowledge DARPA, IARPA, NSF, and ONR for providing partial support of this work.

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2019. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32:690–701.

Eric B Baum. 1988. On the capabilities of multilayer perceptrons. *Journal of complexity*, 4(3):193–215.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

[5] For IWSLT, we follow the pre-processing steps in Edunov et al. (2018). The train/val/test split is 129k/10k/6.8k sentences. For WMT, we follow pre-process as in Ott et al. (2018), with 4.5M/16.5k/3k sentences in train/val/test.

Ondřej 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*, Baltimore, Maryland, USA. Association for Computational Linguistics.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and Marcello Federico. 2015. Report on the 11 th iwslt evaluation campaign , iwslt 2014. In *Proceedings of IWSLT*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Jared Davis, Tamas Sarlos, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. In *International Conference on Learning Representations*.

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*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

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

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR.

Adam Gaier and David Ha. 2019. Weight agnostic neural networks. *arXiv preprint arXiv:1906.04358*.

Claudio Gallicchio and Alessio Micheli. 2017. Echo state property of deep reservoir computing networks. *Cognitive Computation*, 9(3):337–350.

A. Gamba, L. Gamberini, G. Palmieri, and R. Sanna. 1961. Further experiments with papa. *Il Nuovo Cimento (1955-1965)*, 20(2):112–115.

Zhe Gan, Yen-Chun Chen, Linjie Li, Tianlong Chen, Yu Cheng, Shuohang Wang, and Jingjing Liu. 2021. Playing lottery tickets with vision and language. *arXiv preprint arXiv:2104.11832*.

Shankar Iyer, Nikhil Dandekar, and Kornl Csernai. 2017. First quora dataset release: Question pairs, 2017. *URL https://data. quora. com/First-Quora-Dataset-Release-Question-Pairs*.

Herbert Jaeger. 2003. Adaptive nonlinear system identification with echo state networks. In *Advances in neural information processing systems*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning*. PMLR.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mantas Lukoševičius and Herbert Jaeger. 2009. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3).

Wolfgang Maass, Thomas Natschläger, and Henry Markram. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*, 32:14014–14024.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*.

Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. 1994. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Jonathan Pilault, Jaehong Park, and Christopher Pal. 2020. On the impressive performance of randomly weighted encoders in summarization tasks. *arXiv preprint arXiv:2002.09084*.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning. *arXiv preprint arXiv:2005.00561*.

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*.

Ali Rahimi and Benjamin Recht. 2008. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Ali Rahimi and Benjamin Recht. 2009. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. 2020. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11893–11902.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33.

Simone Scardapane and Dianhui Wang. 2017. Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2):e1200.

Wouter F Schmidt, Martin A Kraaijveld, and Robert PW Duin. 1992. Feedforward neural networks with random weights. In *Proceedings of the 11th International Conference on Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems*, pages 1–4.

Sheng Shen, Alexei Baevski, Ari S Morcos, Kurt Keutzer, Michael Auli, and Douwe Kiela. 2021. Reservoir transformers. In *ACL*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170.

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 5998–6008.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. *arXiv preprint arXiv:2003.12298*.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. *arXiv preprint arXiv:1901.10444*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. 2020. Supermasks in superposition for continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 6.

Zhewei Yao, Linjian Ma, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. 2021. Mlpruning: A multilevel structured pruning framework for transformer-based models. *arXiv preprint arXiv:2105.14636*.

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*.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pages 3597–3607.