Evaluating Neural Network Pruning Techniques on Visual Transformers

Sarah Chen
Stanford University
sachen@stanford.edu

Victor Kolev
Stanford University
vkolev@stanford.edu

Kaien Yang
Stanford University
kaieny@stanford.edu

Jonathan Frankle
Mosaic ML
jonathan@mosaicml.com

Abstract

As Transformer models grow ever larger, their efficiency becomes of increasing importance. As such, the field of model compression, specifically network pruning, is of particular relevance. In this work, we rigorously evaluate standard magnitude-based pruning techniques on Vision Transformers (ViTs) to observe ViT-specific behaviour and gain a deeper understanding of the unique properties which contribute to their high performance. Our findings suggest that ViTs can achieve high sparsity ratios without significant accuracy loss, both in structured and unstructured pruning settings. We observe that the effects of pruning vary when applied to different layers, and our results from this analysis highlight the key role played by attention units in Transformers. We also find surprising spatial patterns in the weight matrices yielded by unstructured pruning, which reveal an implicit structure in the underlying model. We observe that fine-tuning after pruning has a strong effect of recovering accuracy, mitigating sub-optimal or one-shot pruning strategies. Lastly, we find that magnitude-based structured pruning only marginally outperforms a random pruning baseline, suggesting a direction for future work.

1. Introduction

Transformer networks have demonstrated state-of-the-art results in many fields of deep learning, and Transformer-based foundation models [3] promise great potential for powering the next wave of deep learning solutions. This stems from the fact that Transformers can attain better performance by increasing parameter count and training time, without suffering from the traditionally expected diminishing returns [24], as well as the fact that they exhibit significantly different scaling behaviours compared to other neural architectures [38, 19].

Simply increasing model size to achieve better results, however, incurs high computational costs, storage requirements, and environmental impacts [2]. It is therefore relevant to consider how to train and deploy models with similar performance but greater efficiency. Approaches that have shown promise in this field are efficient attention [21, 36, 5], model compression [17, 12], and model pruning — the last of which, and its optimal adaptation to Vision Transformer (ViT) models [7], is the principal focus of this paper.

Pruning methods for neural networks have enabled large gains in efficiency (via 10-100x reductions in model size) at little cost to accuracy [18]. Pruning has even been shown to have a regularizing effect in which performance can increase after pruning [31]. However, standard pruning methods have not yet been thoroughly evaluated in the context of ViTs. In fact, pruning work generally suffers from a lack of standard benchmarks. Additionally, although many complex sparsification techniques have been proposed on small datasets, many have failed to generalize to larger-scale experiments, proving less effective than simple magnitude-based pruning methods [11].

To provide a comprehensive benchmark for ViT pruning, we train a small ViT on CIFAR-10 and evaluate a spectrum of magnitude-based train-then-sparsify pruning methods on the model. We consider whether methods that have been shown to be successful on MLPs, Transformers for natural language, and CNNs demonstrate comparable performance on ViTs. We also investigate the shortcomings of these pruning strategies, as well as how we can move towards better pruning strategies for ViTs specifically. Our main contributions are summarized as follows:

- By rigorously benchmarking magnitude-based pruning methods for ViTs, we conclude that ViTs are highly conducive to pruning.
- There is a notable difference between the prunability of different layers, suggesting an uneven distribution of information across layers and across layer types, high-

lighting the importance of attention.

- Weight matrices of the ViT are highly structured, enabling implicit structured pruning to occur in an unstructured setting.
- Fine-tuning strongly mitigates performance degradation, even when one-shot pruning or sub-optimal pruning strategies are used.
- Magnitude-based structured pruning approaches do not significantly outperform random pruning, suggesting the potential for further research into non-standard pruning strategies.

2. Related work

Pruning methods fall into two main categories: (1) unstructured pruning, which removes individual weights [16], and (2) structured pruning, which removes model substructures [4]. These methods traditionally use train-then-sparsify schedules in which the standard dense training is run to convergence and the model is subsequently pruned and retrained with the sparsified structure [18]. Alternative training schedules include sparsify-during-training, in which a model is iteratively sparsified before convergence, and fully-sparse training, where a sparse model is pruned and regrown during training [13, 27, 18].

Hoefler et al. provide an in-depth overview of different pruning techniques, and their work informs the approaches we apply to ViTs [18]. Particularly relevant to our work is the research on pruning applied to transformers and ViTs, specifically. Gale et al. review classic pruning techniques on a large-scale Transformer model for neural machine translation, providing general insights into the behavior of these methods on a Transformer model [11]. Chen et al. explore traditional pruning methods on ViTs, including the one-shot magnitude-based pruning method [15] that we also consider, and propose a sparsify-during-training approach for both structured and unstructured pruning [4].

Work in the context of NLP models has explored pruning entire attention heads [26, 1, 39, 35] or removing intermediate layers [32]. However, these techniques largely rely on removing parts of the network and retraining in a grid-search fashion, and we limit the scope of our research to how more targeted and well-defined pruning techniques translate to ViTs.

We also note another line of work with regards to model compression, namely decomposing matrices via techniques such as SVD and similar low-rank matrix approximations [37, 14], or Kronecker products [34, 8]. While this line of work is promising, its focus is primarily on approximating weight matrices, and as such provides little insight into the learning process of the Transformer and its properties, and as such does not align with our larger goal of moving towards better understanding of the Transformer.

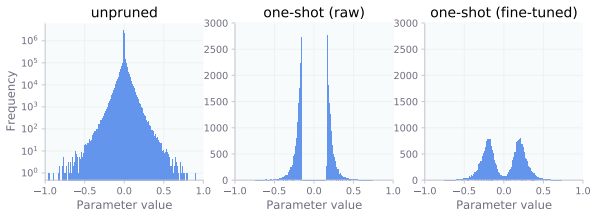


Figure 1: Weight magnitudes for global unstructured pruning at 99.5% sparsity. Model weights are centered around 0 and after pruning and fine-tuning, weights do not return to their original distribution, with weights making peaks around the pruning threshold but not clustering around 0.

Lastly, we note that magnitude-based pruning is far from the only pruning strategy, but it is the one that is most widely used and has proven the most robust and consistent in delivering results [18]. As such, it is the primary focus of our work.

3. Methods

Vision Transformer We anchor our unpruned ViT to an existing open-source baseline [28] implemented in PyTorch [29]. This 6.3-million-parameter network, consisting of 7 layers with 12 attention heads and 384 embedding dimensions, serves as the trained model on which we apply pruning. Since the converged model weights are not provided [28], we train the ViT from scratch over 200 epochs, using the same hyperparameters as the original for maximum reproducibility. Specifically, we use the Adam optimizer [20] with β_1 and β_2 coefficients set to 0.9 and 0.999 respectively, a batch size of 128, and a weight decay of 5×10^{-5}. The learning rate is initialized to 1×10^{-3} and adjusted according to a cosine annealing schedule with a warmup of 5 epochs. AutoAugment [6] and 10% label smoothing [33] serve to regularize the ViT. Training the ViT with our code takes approximately 4 hours on an NVIDIA T4 GPU.

Pruning Methodology We consider 3 pruning methods: (1) unstructured, (2) structured per-row, and (3) structured per-column pruning. For each method, we consider two pruning schedules: (1) one-shot pruning with fine tuning and (2) iterative pruning. For unstructured pruning methods, we consider global (globally pruning minimum magnitude weights, agnostic to layer) and layerwise (pruning a uniform fraction of minimum magnitude for each layer) pruning distributions, in addition to a random baseline. For structured pruning methods, we consider only per-layer pruning distribution and a random baseline, since the differ-

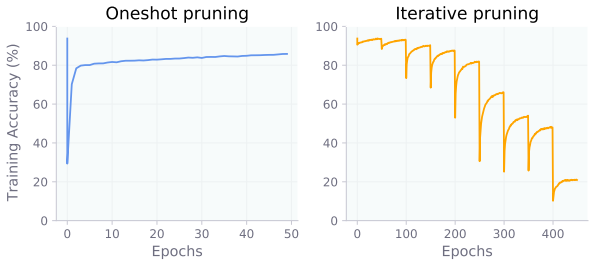


Figure 2: Schematic of training accuracy for the trained model after one-shot pruning with fine-tuning vs. iterative pruning.

ent weight scalings of each layer otherwise lead to certain layers being pruned in their entirety, even for smaller pruning ratios. While there exist ways to circumvent this issue, such as class-distributed pruning, where sparsity ratios per layer are scaled by the standard deviation of the weight matrix, we found this to be beyond the scope of this paper.

Fine-tuning In order to maintain high accuracy it is imperative that the model is fine-tuned after pruning. There has been considerable literature on the topic [25, 23], and we follow the approach of learning rate rewinding, as suggested by Renda et al. [30] — retraining for 25% of the original training time (within the rewinding safe zone [30]) and, notably, rewinding the learning rate schedule to its value before the last 25% of the training schedule.

Experiment Configurations We outline three main experimental setups, through the combination of which all experiments can be derived.

- *Unstructured one-shot magnitude-based pruning* removes the $p\%$ of the weights with smallest magnitude after training. This pruning method updates the weights of a model in the manner shown in Figure 1, for example, which was generated by pruning 99.5% of weights. We follow pruning with fine-tuning, as illustrated in Figure 2.

  More formally, for global pruning, for a network $f(x;\theta)$ with parameters $\theta \in \mathbb{R}^{n_\theta}$ we derive a mask $\mu \in \{0,1\}^{n_\theta}$ on the parameters such that

$$\mu = \operatorname{argmin} \|(1-\mu)\odot\theta\|_1 \quad \text{s.t.} \quad \frac{\|1-\mu\|_1}{n_\theta} = r,$$

  where r is the desired sparsity ratio and $\odot$ is the Hadamard product. The network then continues to fine-tuning as $f(x;\mu\odot\theta)$.

  Similarly, for layerwise pruning on a layer with weight matrix $W \in \mathbb{R}^{m\times n}$, we derive a mask $\mu_W \in \{0,1\}^{m\times n}$ such that

$$\mu_W = \operatorname{argmin} \|(1-\mu_W)\odot W\|_1 \quad \text{s.t.} \quad \frac{\|1-\mu_W\|_1}{mn} = r.$$

- *Structured one-shot magnitude-based pruning* removes entire rows or columns of a weight matrix based on their l_1 norm. Note that removing a row sets all incoming weights to a neuron to zero, and removing a column removes all outgoing weights from an neuron. In either approach, the neuron is removed; what differs is the criteria to determine which neurons to remove. It is notable that structured pruning allows us to achieve tangible gains in computational efficiency during inference, since it directly reduces the weight matrices.

  Formally, considering a weight matrix $W \in \mathbb{R}^{m\times n}$, for per-row structured pruning we consider a mask $\mu_W \in \{0,1\}^m$ such that

$$\mu_W = \operatorname{argmin} (1-\mu_W \mathbf{1}_n^\top)\odot W \quad \text{s.t.} \quad \frac{\|1-\mu_W\|_1}{m} = r,$$

  where $\mathbf{1}_n$ is an n-dimensional vector of ones. Similarly, for per-column pruning we have $\mu_W \in \{0,1\}^n$ such that

$$\mu_W = \operatorname{argmin} (1-(\mu_W \mathbf{1}_m^\top))\odot W^\top \quad \text{s.t.} \quad \frac{\|1-\mu_W\|_1}{n} = r.$$

- *Iterative pruning* We evaluate iterative versions of the above pruning methods by utilizing an iterative pruning schedule, which entails pruning in smaller steps until the desired sparsity ratio and fine-tuning between consecutive steps to minimize performance loss. (Figure 2).

Evaluation We evaluate the outlined magnitude-based train-then-sparsify pruning methods on the trained model. All linear layers within the model are eligible for pruning[1]. To evaluate each pruning strategy, we compare validation accuracy against different sparsity ratios. This performance curve further informs the extent of over-parameterization in the model. We also perform a qualitative evaluation of the methods by visualizing masks, sparsity ratio distributions across modules, weight distributions, etc. To serve

[1]We excluded the input embedding layer for structured per-row pruning and the feedforward output layer for structured per-column pruning to avoid excluding entire input channels or entire output classes.

as a baseline, we perform random pruning by shuffling the masks yielded by magnitude-based pruning for each layer. This baseline illustrates the accuracy that can be trivially achieved by a given sparsity ratio. Notably, we preserve the per-layer sparsity ratios of the original masks rather than determining those ratios randomly. This distinction allows us to attribute any observed improvements to the pruning methods specifically, with no dependence on per-layer sparsities.

Implementation We drew from the OpenLTH GitHub repository to support our pruning experiments [9]. In particular, we used the Mask class and related functions that determine prunable layers and apply masks, either in their entirety or with minor modifications. We took heavy inspiration from their `sparse_global` pruning method and random shuffling methods, which implement global unstructured pruning, but from that point we produced our own implementations of all pruning strategies, alongside a module experimental framework for all pruning schedules, methods, and distributions.[2]

Dataset CIFAR-10 is a small-scale image classification dataset consisting of 60,000 colored images evenly distributed across 10 classes, with 50,000 training observations and 10,000 testing observations [22]. Each image has a fixed size of 32×32 pixels. We preprocess the data by normalizing each of the three color channels and performing data augmentation as described in [6].

4. Results

Vision Transformer The unpruned ViT achieves 90% test set accuracy on CIFAR-10 after 200 epochs, as shown in Figure 3. This indicates that training iterations are fast and computationally inexpensive. Our results are consistent with those reported in [28]. From this base model, we determined the relationship between sparsity and performance for each pruning strategy (Figure 4).

Unstructured Pruning Pruned models sustain high accuracy even at high pruning ratios. In particular, global unstructured pruning achieves high sparsity without seeing performance degradation. Even at 95% sparsity, we observe only a 1.54% drop in validation accuracy, from 90.44% to 88.90%, while random pruning drops to 59.61%. That the model sustains high performance at 95% accuracy reflects existing benchmarking results for CIFAR-10 on CNNs like VGG-19 [25]. Layerwise unstructured pruning maintains similarly strong performance until 97.5% sparsity, but, consistent with past findings on NMT models, global magnitude-based pruning is more effective than the layerwise equivalent [31].

[2]Code for our experiments can be found on https://github.com/sarahlc888/vit-pruning.

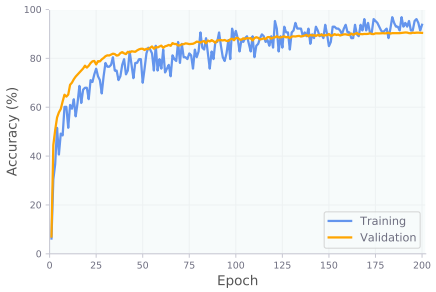


Figure 3: Accuracy curve for the unpruned ViT. We achieve a final accuracy of 90.44%, in line with results in [28].

Structured Pruning Similarly to NMT models, ViTs are less conducive to structured pruning. Although the pruning strategy outperforms the random equivalent, its accuracy drops below 85% upon reaching 90% sparsity for both per-column and per-row, and it only outperforms the random baseline accuracy by approximately 10% upon reaching 98.5% sparsity. Using structured pruning at such high sparsity ratios is more challenging than unstructured approaches due to the removal of entire rows and columns rather than individual elements of a weight matrix. It is further noteworthy that structured per-row and per-column pruning do not significantly differ in the results, suggesting that the model can adapt to both variants.

Pruning Patterns We find that that structured trends emerge on the per-layer level when applying global unstructured pruning (Figure 6). Even at low pruning ratios, entire layers are pruned, revealing that their weights have smaller magnitude than those in other matrices. For example, the key and query attention matrices in layer 0 as well as select MLP matrices in layers 4 and 5 are pruned to nearly 100% sparsity, even at the relatively low pruning ratios of 25% and 50%.

Structured trends also emerge within each layer's weight matrices, on the per-row and per-column level, especially for attention layers, where we see the same rows being pruned across keys, values, and queries, as well as whole attention heads being pruned together (figures in Appendix A). Similarly, in the structured pruning case, whole attention heads are implicitly pruned across query, key, and value matrices, an effect which is especially visible in layer 4.

Chen et al. also observe per-column structural patterns within weight matrices for unstructured methods in their work on pruning ViTs [4]. The emergence of these structural patterns suggests that even unstructured methods may

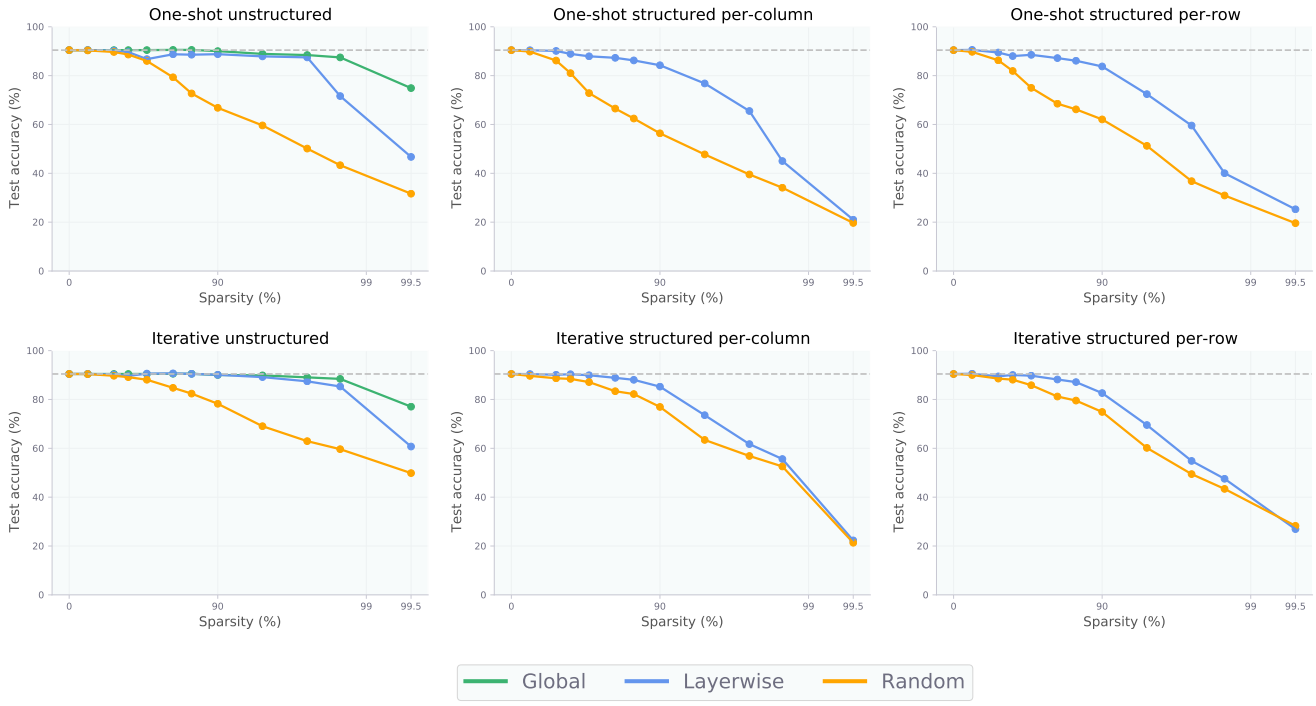


Figure 4: Model performance after one-shot pruning with fine-tuning (top) and iterative pruning (bottom). Each column is the pruning *method*: unstructured, structured per-row, and structured per-column. On each subplot, all pruning is ℓ_1 magnitude-based with *distribution* variants: global and layerwise, as well as a random baseline. We find that the magnitude-based methods consistently outperform their respective random baselines. Iterative varieties on average tend to perform better than their one-shot counterparts, yet all methods show high resilience to pruning.

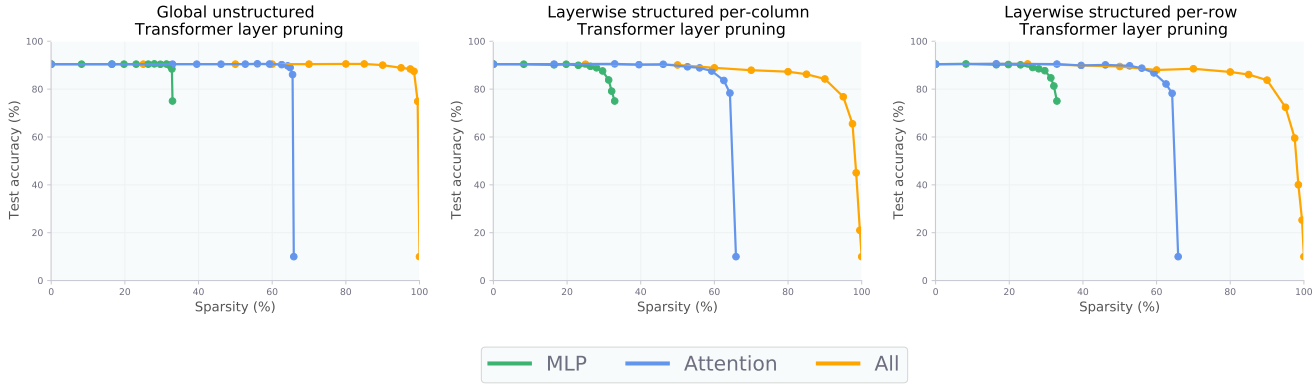


Figure 5: Model performance when one-shot pruning is applied to MLP layers only, attention layers only, and all layers. For every experiment, we use a set of sparsity ratios up to 100%.

offer possibilities for increased hardware efficiency.

Iterative Pruning We find that iterative pruning provides a surprisingly small benefit for the $n-$fold increase in computation time, where n is the number of steps. Moreover, we find that masks are very similar between one-shot and iterative pruning strategies (Figure 7 in Appendix A). This suggests that fine-tuning (or, more precisely, learning rate rewinding) can recover accuracy very efficiently, even though training a similarly sized ViT as a sparse model to

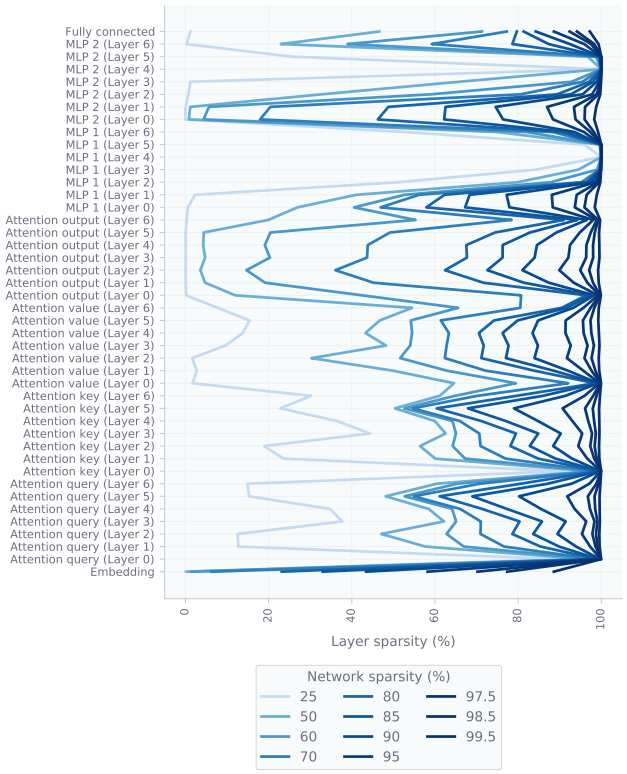


Figure 6: Per-layer weight matrix sparsities generated by one-shot global unstructured pruning. For each pruning ratio, the observed per-layer sparsity varies across layer numbers and types.

the same accuracy would be intractable. This hints that the Transformer needs to be overparametrized to train well, but can operate in a highly sparse regime at inference time.

Analysis of Pruning on Network Modules We analyse the pruning ratios for each layer (Figure 6). For global unstructured pruning, we see that different layers consistently get pruned at different ratios: specifically, the fully connected and embedding layers retain more parameters than other layers; earlier layers benefit from a higher density of weights in the feed-forward layers, while later layers prioritize the attention layers, with the MLP layers 4, 5 being almost entirely removed. This suggests that the Transformer optimizes a embedding/pre-process $\rightarrow$ attention $\rightarrow$ post-process/output pipeline.

Transformer-specific Pruning To test the relative importance of feed-forward layers and attention layers, we compare pruning only MLP layers and pruning only from attention layers. As per Figure 5, even when pruning all the MLP layers, relying only on residual connections between attention, the network is able to maintain reasonable accuracy, whereas if we prune all the attention layers, accuracy drops down to random guessing. This highlights the fact that attention is paramount in facilitating the performance of the Transformer architecture, in line with other studies [7].

5. Conclusion & Future Work

Our findings suggest that ViTs can sustain high sparsity levels, especially for global unstructured pruning methods. We observe the phenomenon that Transformers exhibit internal structure that enables implicit structured pruning in an unstructured global pruning objective. Additionally, we emphasize the importance of fine-tuning, as it has a strong effect on recovering accuracy after pruning, even when working with sub-optimal pruning strategies.

Our work confirms some intuitions and widely held beliefs about Transformers more broadly, namely that attention mechanisms are principal in attaining good performance. Moreover, overparameterizing the network for training enables attaining very high performance, yet during inference few of these parameters are actually needed – we hypothesize that there is a fundamental difference between the number of parameters needed for training and the number of parameters used in a trained model, i.e. training requires a larger network capacity, which is also in support of other works [10, 38, 19].

Lastly, we find that there is much room for improvement in the case of structured pruning, which is a potential direction for future work.

6. Contributions & Acknowledgements

We would like to thank Jonathan Frankle for proposing the project idea and providing guidance on the landscape of pruning literature, experiments of interest, and interpretation of results, as well as for introducing us to the field.

We made use of code from ViT-CIFAR [28] and OpenLTH [9], as described in Methods. Our codebase uses the `torch`, `numpy`, `matplotlib` python libraries.

Among the authors, conceptual work was completed in group discussion. Kaien Yang focused on training the ViT while Sarah Chen and Victor Kolev focused on pruning method implementation. All contributed to running experiments and analyzing results.

References

[1] Maximiliana Behnke and Kenneth Heafield. Losing heads in the lottery: Pruning transformer attention in neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2664–2674, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.211. URL https://aclanthology.org/2020.emnlp-main.211.

[2] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[4] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

[5] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Kronecker decomposition for gpt compression. *arXiv preprint arXiv:2110.08152*, 2021.

[9] facebookresearch. Openlth. https://github.com/facebookresearch/open_lth, 2020.

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

[11] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[12] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 2021.

[13] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

[14] Habib Hajimolahoseini, Mehdi Rezagholizadeh, Vahid Partovinia, Marzieh Tahaei, Omar Mohamed Awad, and Yang Liu. Compressing pre-trained language models using progressive low rank decomposition.

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[17] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[18] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[19] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Duong H Le and Binh-Son Hua. Network pruning that matters: A case study on retraining variants. *arXiv preprint arXiv:2105.03193*, 2021.

[24] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *arXiv preprint arXiv:2002.11794*, 2020.

[25] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[26] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

[27] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[28] Omihub777. Vit-cifar. https://github.com/omihub777/ViT-CIFAR, 2021.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[31] Abigail See, Minh-Thang Luong, and Christopher D Manning. Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*, 2016.

[32] Sharath Nittur Sridhar and Anthony Sarah. Undivided attention: Are intermediate layers necessary for bert? *arXiv preprint arXiv:2012.11881*, 2020.

[33] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[34] Marzieh S Tahaei, Ella Charlaix, Vahid Partovi Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. Kroneckerbert: Learning kronecker decomposition for pre-trained language models via knowledge distillation. *arXiv preprint arXiv:2109.06243*, 2021.

[35] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

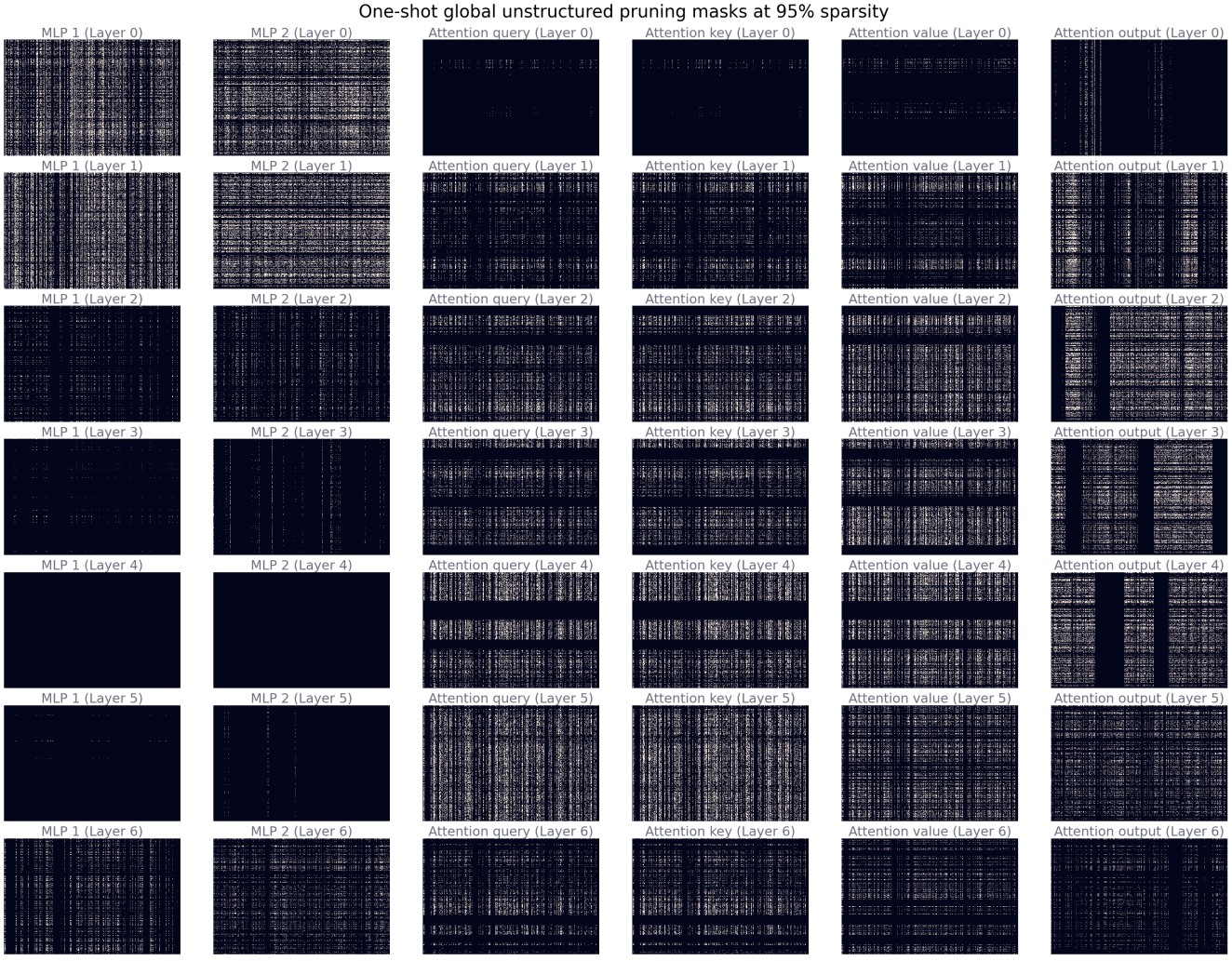
[36] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[37] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017.

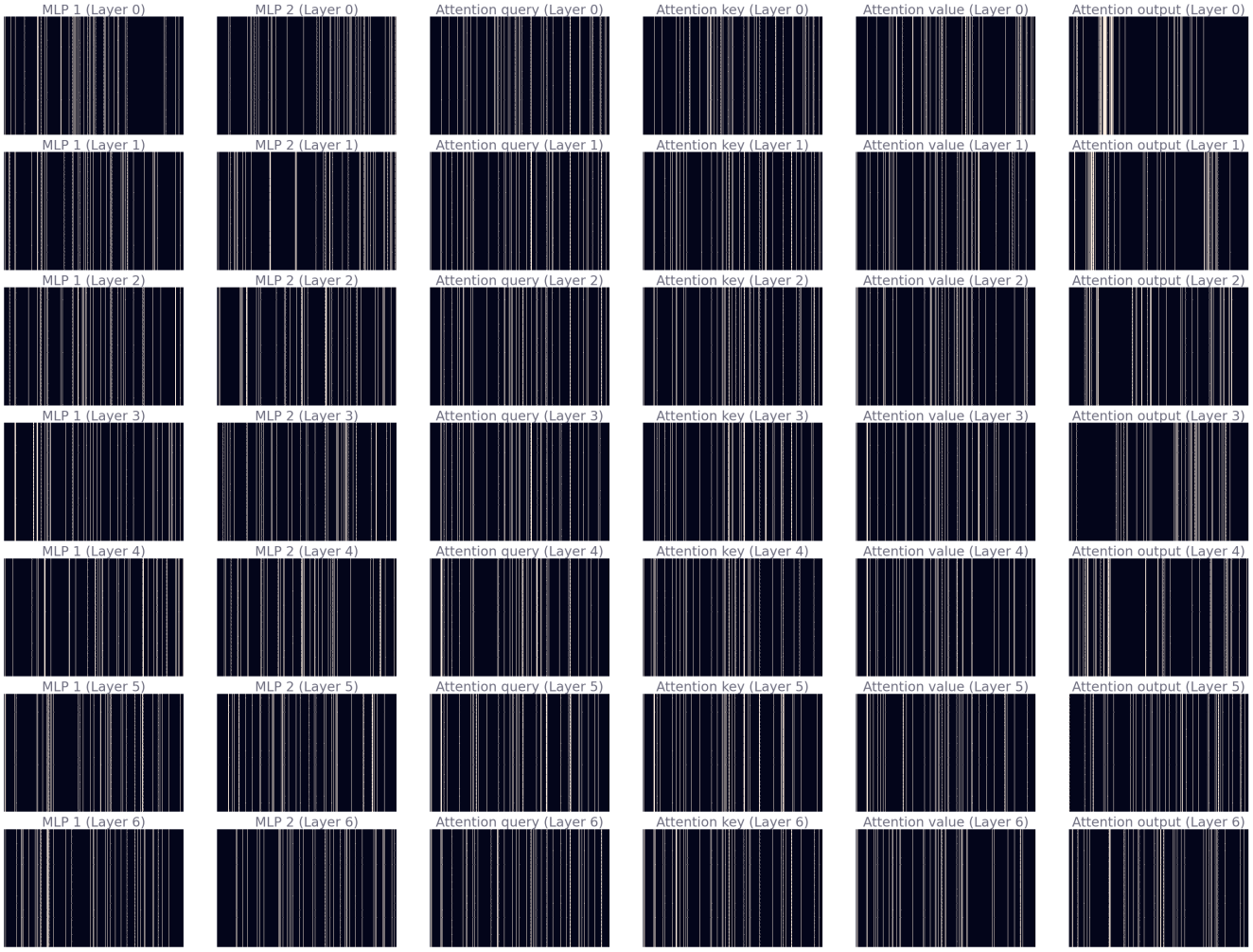
[38] X Zhai, A Kolesnikov, N Houlsby, and L Beyer. Scaling vision transformers. arxiv 2021. *arXiv preprint arXiv:2106.04560*.

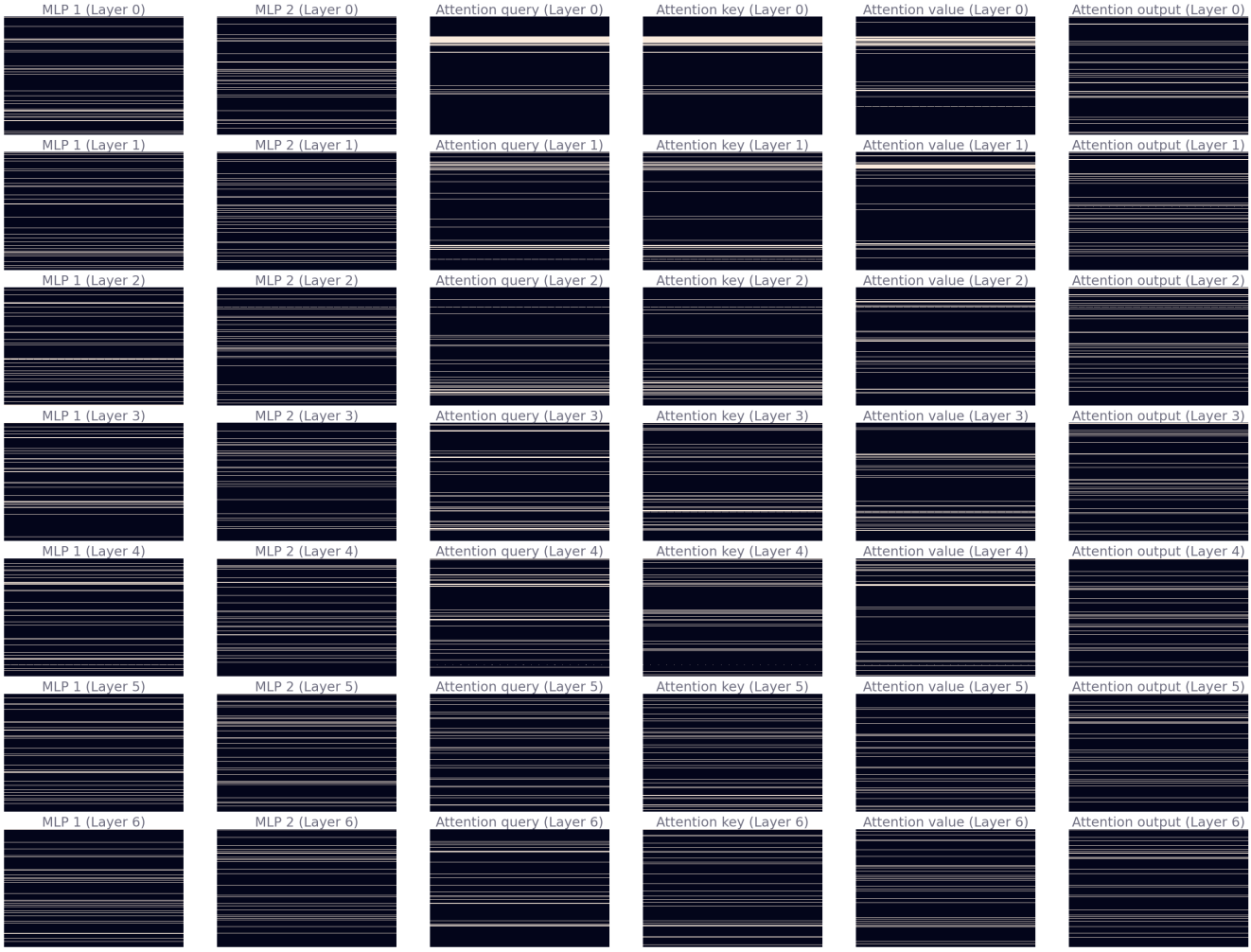
[39] Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. Know what you don't need: Single-shot meta-pruning for attention heads. *AI Open*, 2:36–42, 2021. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.05.003. URL https://www.sciencedirect.com/science/article/pii/S2666651021000140.

Appendix A. Mask visualisations & similarity


One-shot global unstructured pruning masks at 95% sparsity
MLP 1 (Layer 0)
MLP 2 (Layer 0)
Attention query (Layer 0)
Attention key (Layer 0)
Attention value (Layer 0)
Attention output (Layer 0)
MLP 1 (Layer 1)
MLP 2 (Layer 1)
Attention query (Layer 1)
Attention key (Layer 1)
Attention value (Layer 1)
Attention output (Layer 1)
MLP 1 (Layer 2)
MLP 2 (Layer 2)
Attention query (Layer 2)
Attention key (Layer 2)
Attention value (Layer 2)
Attention output (Layer 2)
MLP 1 (Layer 3)
MLP 2 (Layer 3)
Attention query (Layer 3)
Attention key (Layer 3)
Attention value (Layer 3)
Attention output (Layer 3)
MLP 1 (Layer 4)
MLP 2 (Layer 4)
Attention query (Layer 4)
Attention key (Layer 4)
Attention value (Layer 4)
Attention output (Layer 4)
MLP 1 (Layer 5)
MLP 2 (Layer 5)
Attention query (Layer 5)
Attention key (Layer 5)
Attention value (Layer 5)
Attention output (Layer 5)
MLP 1 (Layer 6)
MLP 2 (Layer 6)
Attention query (Layer 6)
Attention key (Layer 6)
Attention value (Layer 6)
Attention output (Layer 6)

One-shot layerwise structured per-column pruning masks at 95% sparsity




One-shot layerwise structured per-row pruning masks at 95% sparsity
MLP 1 (Layer 0)
MLP 2 (Layer 0)
Attention query (Layer 0)
Attention key (Layer 0)
Attention value (Layer 0)
Attention output (Layer 0)
MLP 1 (Layer 1)
MLP 2 (Layer 1)
Attention query (Layer 1)
Attention key (Layer 1)
Attention value (Layer 1)
Attention output (Layer 1)
MLP 1 (Layer 2)
MLP 2 (Layer 2)
Attention query (Layer 2)
Attention key (Layer 2)
Attention value (Layer 2)
Attention output (Layer 2)
MLP 1 (Layer 3)
MLP 2 (Layer 3)
Attention query (Layer 3)
Attention key (Layer 3)
Attention value (Layer 3)
Attention output (Layer 3)
MLP 1 (Layer 4)
MLP 2 (Layer 4)
Attention query (Layer 4)
Attention key (Layer 4)
Attention value (Layer 4)
Attention output (Layer 4)
MLP 1 (Layer 5)
MLP 2 (Layer 5)
Attention query (Layer 5)
Attention key (Layer 5)
Attention value (Layer 5)
Attention output (Layer 5)
MLP 1 (Layer 6)
MLP 2 (Layer 6)
Attention query (Layer 6)
Attention key (Layer 6)
Attention value (Layer 6)
Attention output (Layer 6)

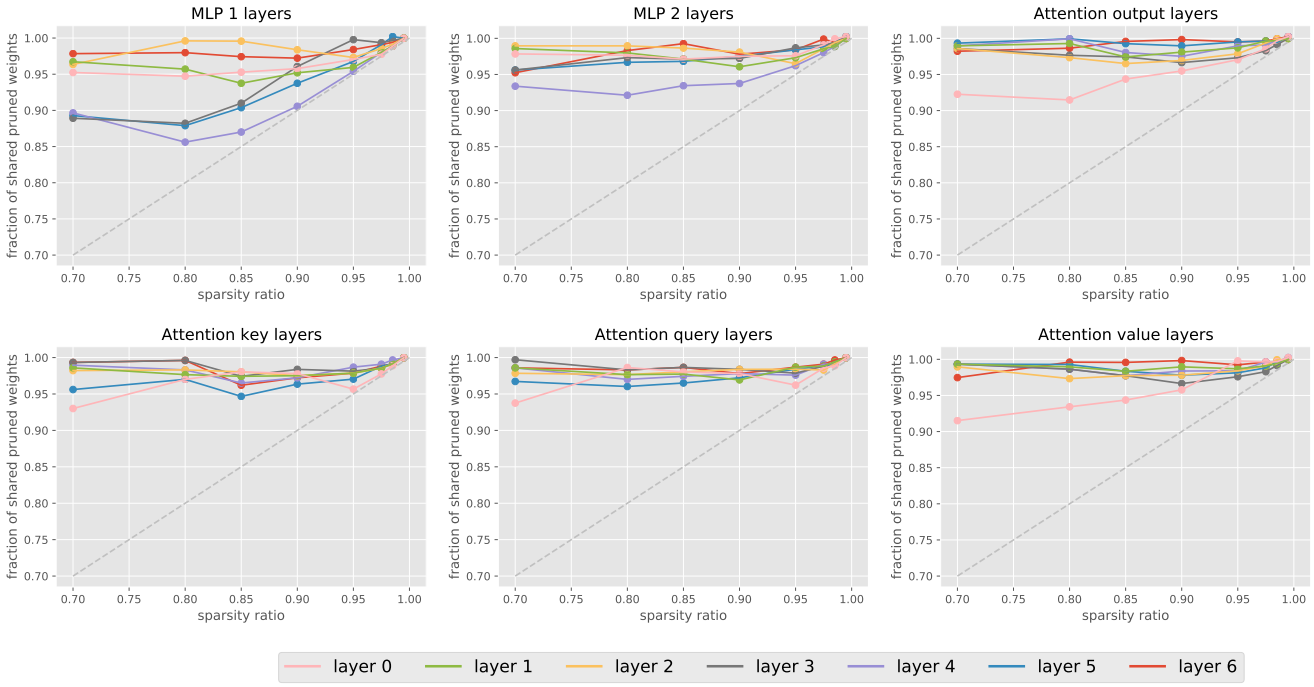


Figure 7: Per-layer mask similarities between one-shot and iterative pruning for l_1 structured magnitude-based pruning. Similarity is calculated as the fraction of weights that both one-shot and iterative pruning mask out for a given sparsity. The figure begins at 70% sparsity, the point at which the pruned model fails to sustain the original unpruned accuracy.