The Unreasonable Ineffectiveness of the Deeper Layers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Understanding *where* and *how* knowledge is stored in LLMs is an active and important area of research. In this work, we take a model pruning approach: if removing certain parameters does not affect model output in question-answering knowledge benchmarks, then those parameters are likely are not useful for storing knowledge. To find these parameters, we design simple layer-pruning strategies for popular families of open-weight pretrained LLMs, finding minimal degradation of performance on different question-answering benchmarks until after a large fraction (up to half) of the layers are removed. Concretely, we identify the optimal block of layers to prune by considering similarity across layers; then, to "heal" the damage, we perform a small amount of finetuning. From a scientific perspective, the robustness of these LLMs to the deletion of layers implies either that current pretraining methods are not properly leveraging the parameters in the deeper layers of the network or that the shallow layers play a critical role in storing knowledge.

1 Introduction

Over the last few years, large language models (LLMs) have evolved from mere research artifacts [1] into useful products [2]. As language model abilities improve [3, 4] and they are used more widely, it becomes increasingly important to understand *how* language models store knowledge internally (one can imagine being able to update incorrect knowledge in LLMs directly). This question is commonly approached through interpretability studies, which produce post-hoc explanation of what certain parameters are doing, for example by probing internal model representations on specific tasks [5–7], or analyzing model activations [8, 9] and finding "circuits" responsible for certain behaviors [10, 11]. Ideally, one would go further than interpreting model representations, and directly intervene to control model behavior. While some studies have attempted to use their mechanistic understanding to edit world knowledge stored in models [12], subsequent work demonstrates that these methods and knowledge localization may be uncorrelated [13].

We propose using model pruning as a framework for understanding open-weight LLMs — model pruning emphasizes finding subsets of parameters that can be removed without affecting model performance. This serves as a suitable intervention for understanding how a network uses its parameters: if you can remove sections of a network with minimal effect on its performance, then those parameters are likely not important to your specific task. Moreover, using model pruning as an intervention for understanding leads to practical results, as at the end of your investigation you actually obtain a smaller model that performs well on your task. We design very simple layer pruning strategies using open-weight LLMs and measure performance degradation on common question-answering benchmarks. Our method uses the similarity between the representations at different layers to identify the optimal layers to prune for a given pruning fraction; then, after removing these layers we "heal" the pruning-induced mismatch with a small amount of fine tuning (using QLoRA). Our main empirical result is that we can remove a substantial fraction of the *deepest layers* from models with minimal degradation in performance on QA benchmarks. For example, for Llama-2-70B [14]

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

we can eliminate up to roughly ***half*** of the layers before the performance collapses on MMLU. The robustness of open-weight LLMs to removal of deeper layers and the sharp transition in performance on downstream knowledge tasks (e.g. MMLU and BoolQ) suggest that shallow layers may play a critical role in storing knowledge.

2 Layer-pruning algorithm(s)

Our principal layer pruning algorithm is very simple:

0. Pick a a number of layers to prune n.
1. Compute the angular distance $d(x^{(\ell)}, x^{(\ell+n)})$, cf. (2) below, between the input to layer ℓ and the input to layer $\ell+n$ on a neutral pretraining dataset or on a dataset representative of a downstream task of interest.
2. Find the layer, ℓ^*, that minimizes that distance:
$$\ell^\star(n) \equiv \arg\min_{\ell} \; d(x^{(\ell)}, x^{(\ell+n)}) \,. \tag{1}$$
3. Drop layers $\ell^\star$ to $\ell^\star{+}n{-}1$; connect the old input to layer $\ell^\star$ to the old $(\ell^\star{+}n)$th layer block.
4. (Optionally) heal the mismatch at layer $\ell^\star + n$ with a small amount of fine tuning on a neutral pretraining dataset or particular dataset of interest.

This algorithm is also pictorally depicted in panels (a)-(d) of Figure 4. The angular distance d in step (1) on a single sequence of length T is given by

$$d(x^{(\ell)}, x^{(\ell+n)}) \equiv \frac{1}{\pi} \arccos\left(\frac{x_T^{(\ell)} \cdot x_T^{(\ell+n)}}{\left\|x_T^{(\ell)}\right\| \left\|x_T^{(\ell+n)}\right\|}\right) , \tag{2}$$

where the inner product is over the hidden dimension of the model for the final token T of the sequence, $\|\cdot\|$ denotes the L^2-norm, and the factor of $1/\pi$ is a convention.

3 Results

For our experiments, we prune a wide variety of large-scale LLMs from 2.7B to 70B parameters spanning 32 to 80 total unpruned layers. Specifically, we used models in the Llama-2 family [14], the Qwen family [15], Mistral-7B [16], and Phi-2 [17]. For these models, we executed the "healing" step using QLoRA [18]: our models were quantized to 4-bit precision and then finetuned, using QLoRA for efficient training, on either 164M or 328M tokens from the Colossal Clean Crawled Corpus (C4) [19], a common pretraining dataset. As a result, *each experiment of ours was performed on a single A*100 *GPU*. For our QA evals, we used Massive Multitask Language Understanding (MMLU) [20], and BoolQ [21] The specifics of our models, healing procedure, dataset choices, and evaluation details can be found across Appendix D; ablations of different hyperparameter choices can be found across Appendix E.

3.1 Pruning as a lens into knowledge localization: accuracy on QA benchmarks

In Figure 1 we show the performance on algorithm described in §2 on MMLU performance. We observe a characteristic flat region of robust performance followed by a sharp transition to random accuracy at a pruning fraction around 45%-55% for models in the Llama-2 family, 35% for Mistral 7B, 25% for Phi-2, and 20% for models from the Qwen family. This implies that the essential knowledge required to achieve a model's top score isn't removed by significant layer removal – even though the fraction can be quite large(!) – until eventually that knowledge is lost at a critical model-dependent threshold. Contrasting the curves with and without healing, we see that finetuning offers a modest improvement by better preserving the unpruned performance and pushing the phase transition to random guessing to slightly larger pruning fractions. Broadly we see that layer pruning is more robust for the larger and deeper models, e.g. Llama-2-13B and Llama-2-70B, which we hypothesize could be related to the fact that either the smaller models are more overtrained, making parameters less redundant, or that the deeper models can afford to lose more layers in an absolute sense. Also, the Qwen family is strange, a fact we will further elaborate on in §3.3.

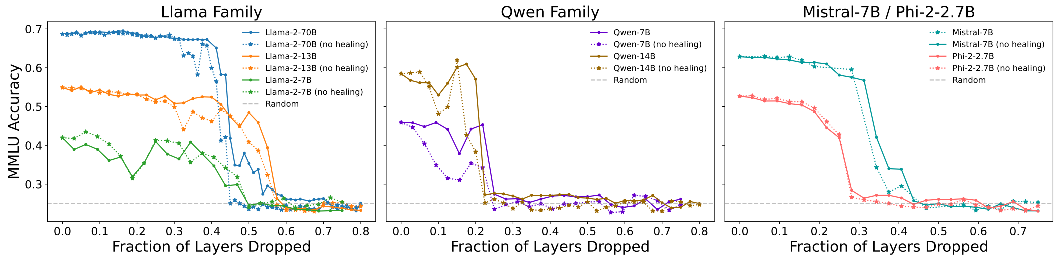

Figure 1: MMLU accuracy (5-shot) vs. fraction of layers dropped for different model families. (*Left:* Llama-2 family; *Middle:* Qwen family; *Right:* Mistral-7B and Phi-2.) The solid lines represent performance after dropping layers and healing, dotted lines show performance after dropping layers only (no healing), and the dashed gray line is the score for guessing randomly. For these models, healing leads to modest improvements, and performances are quite robust until 20%-55% pruning fractions, depending on model family and size, at which point they transitions to random guessing.

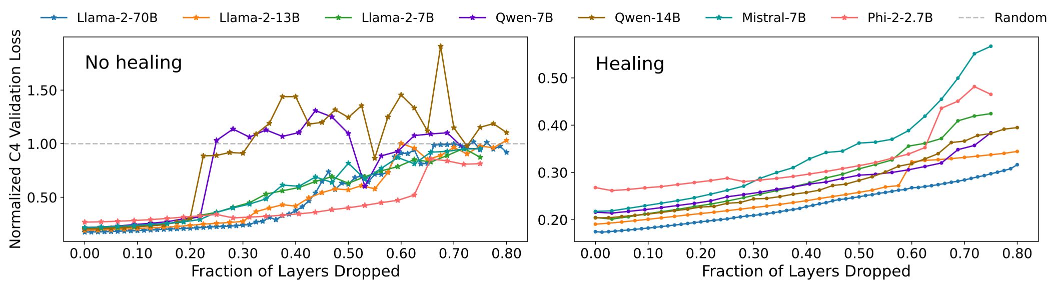

Figure 2: Normalized C4 validation loss vs. fraction of layers dropped before healing (*left*) and after healing (*right*); each curve is normalized by the cross-entropy loss of sampling uniformly from the model's vocabulary. For the experiments before healing, the loss for each model transitions to random guessing (gray dashed line) at approximately the same pruning fractions that the QA benchmarks transition to random guessing; after healing, there is continuity through the regions of sharp transition on QA tasks, cf. Figure 1. Contrasting the overall scale of both plots, it's clear that healing significantly restores the performance on next-token prediction to near-unpruned levels.

3.2 What are deeper layers doing? Analyzing loss on next-token predictions

In Figure 2 , we plot the normalized C4 validation loss for all seven of our models, after healing (left panel) and before healing (right panel). Without healing, we see that there is a somewhat sharp transition to random guessing for each model at approximately the pruning fraction that the QA benchmark accuracies also sharply transition to random guessing (see Figure 1). Contrasting the scales of both plots, we see that healing significantly restores the next-token prediction ability of all the models to near-unpruned levels, with the loss increasing slowly and linearly with layer dropping. This smooth increase in loss highlights *(i)* one way of disconnecting performance on downstream tasks and continuous measures such as cross entropy loss and *(ii)* that deeper layers may be used for some other ability that is learned during pre-training. Preliminary results show that one of these abilities may be reasoning — we evaluate our pruning strategy on GSM8k in Figure 6, and observe that performance immediately drops, suggesting that deeper layers may be important for reasoning.

3.3 Angular distances between representations and a simpler pruning strategy

Given the central role the angular distance (2) plays in our pruning strategy, we analyze these distances across our seven models. For this analysis, the angular distances for each model were averaged over 10k samples from the C4 validation set. In Figure 3 each square is colored to depict the row-normalized angular distance between layer ℓ and $\ell + n$ across all possible ℓ, and n up to very large fractions of the total number of layers; the optimal layer to prune for a given block size, $\ell^*(n)$, corresponds to the minimal distance in each row. Across models, we make two observations: *(i)* deeper layers are typically quite similar to each other and can be more easily dropped; *(ii)* the distances across the blocks that include the last layer are nearly maximal i.e. one should never drop

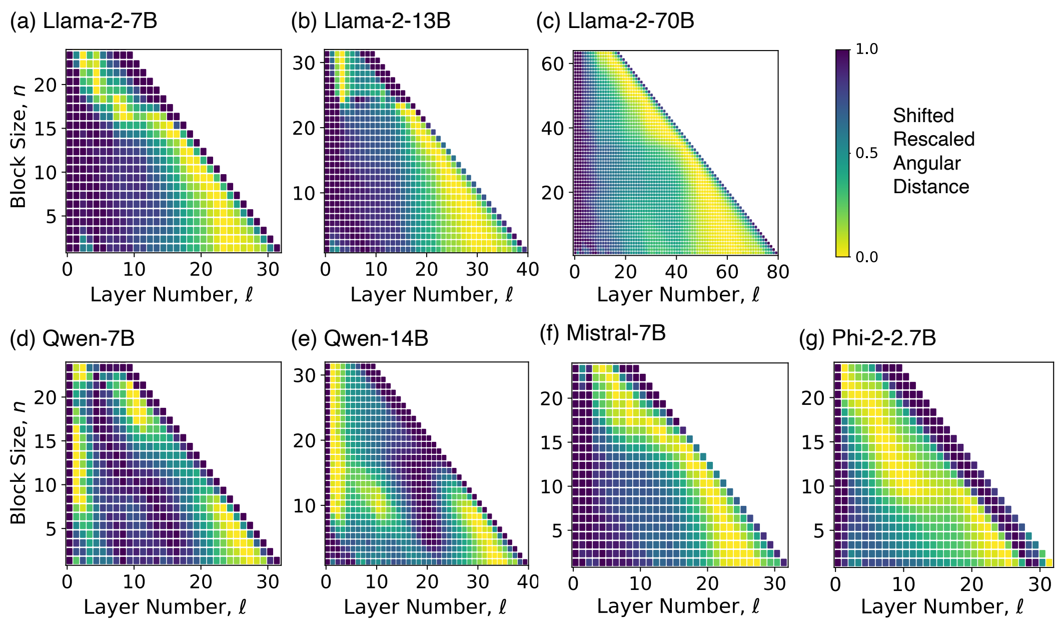

Figure 3: Normalized angular distance (2) from initial layer ℓ (x-axis) with block size n (y-axis) for each of the seven models we evaluated; the distance for each n is shifted and rescaled to span the same range, $[0, 1]$ (yellow to purple): the optimal block to prune, $\ell^*(n)$, corresponds to the deepest yellow for each row. Across models, the deeper layers tend to be very similar, though the deepest blocks that include the final layer (squares along the outer diagonal) are (near-)maximally dissimilar.

the final layer. Inspired by this, we experiment with a very simple heuristic pruning strategy: *(1)* if pruning n layers from an L-layer model, drop layers $(L - n)$ to $(L - 1)$ so as to remove the deepest block that excludes the final layer; then *(2)* heal with a small amount of finetuning as before. This provides a meaningful ablation of the importance of optimizing the block to prune. In Figure 5, we find that this simple heuristic performs poorly without healing the damage incurred by pruning: accuracy on QA benchmarks decays rapidly to (near-) random with increased pruning fraction, and loss begins to increase very rapidly even with small amounts of pruning. However, after healing, the two pruning strategies are quite comparable: for QA benchmarks, the similarity-informed algorithm slightly better preserves the accuracy before the phase transition, though the simple algorithm pushes the phase transition to slightly greater pruning fractions; and for C4 loss, the curves nearly overlap, although the similarity-informed strategy does marginally outperform for all amounts of pruning. These experiments are strong evidence that the purpose of post-pruning finetuning is the healing of damage at the pruning interface and not the acquisition of additional knowledge.

4 Discussion and Future Directions

We leverage model pruning as a tool to understand how open-weight LLMs store knowledge, and demonstrate that we can prune a significant portion (up to 50%) of deeper layers with minimal impact on QA benchmark performance. At the conclusion of the work, we are left with numerous questions: Why does healing eliminate the phase transition in the loss but not in the QA accuracies? With more comprehensive evals, will accuracy on different tasks degrade at different depths? Do pretraining details affect the ability to prune, e.g., are scaling-law over-trained or distilled models more effectively using deeper layers? Some of these questions would benefit from studying both layer similarity and pruning across different pretraining checkpoints; for instance, at what point does the sharp phase transition and critical depth in the QA accuracies emerge, and does more training lead to better use of the later layers? Others suggest explorations with different pretraining architectures and objectives, e.g. in order better make use of the deeper layers. With more comprehensive evaluations, if different kinds of tasks degrade at very different depths, then this might indicate that the knowledge required to complete those tasks is stored at different depths.[1]

[1]Alternatively, one could measure $d(x^{(\ell)}, x^{(\ell+n)})$ or find $\ell^*(n)$ as a function of different eval datasets.

References

[1] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[2] OpenAI. Introducing chatgpt, Nov 2022. URL https://openai.com/blog/chatgpt.

[3] OpenAI. Gpt-4 technical report, 2023.

[4] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[5] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

[6] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

[7] Kevin Clark. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

[8] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

[9] Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? *arXiv preprint arXiv:2310.17191*, 2023.

[10] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[11] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

[12] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

[13] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[14] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[15] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[16] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[17] Mojan Javaheripi and Sébastien Bubeck. Phi-2: The surprising power of small language models, Dec 2023.

[18] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[21] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[22] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

[23] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[24] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.

[25] 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.

[26] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International conference on machine learning*, pages 2285–2294. PMLR, 2015.

[27] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[28] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[29] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[30] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[31] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[32] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2752–2761, 2018.

[33] Kenton Murray and David Chiang. Auto-sizing neural networks: With applications to n-gram language models. *arXiv preprint arXiv:1508.05051*, 2015.

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

[35] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[37] 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.

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

[39] Young Jin Kim and Hany Hassan Awadalla. Fastformers: Highly efficient transformer models for natural language understanding. *arXiv preprint arXiv:2010.13382*, 2020.

[40] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

[41] Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in Neural Information Processing Systems*, 33: 14011–14023, 2020.

[42] Chun Fan, Jiwei Li, Xiang Ao, Fei Wu, Yuxian Meng, and Xiaofei Sun. Layer-wise model pruning based on mutual information. *arXiv preprint arXiv:2108.12594*, 2021.

[43] Ananya Harsh Jha, Dirk Groeneveld, Emma Strubell, and Iz Beltagy. Large language model distillation doesn't need a teacher. *arXiv preprint arXiv:2305.14864*, 2023.

[44] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023.

[45] Wei Liu, Zhiyuan Peng, and Tan Lee. Comflp: Correlation measure based fast search on asr layer pruning. *arXiv preprint arXiv:2309.11768*, 2023.

[46] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33: 9782–9793, 2020.

[47] Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*, 2023.

[48] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

[49] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

[50] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*, 2021.

[51] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[52] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198*, 2023.

[53] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[54] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[55] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[56] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

[57] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[58] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help. *arXiv preprint arXiv:2108.13487*, 2021.

[59] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[60] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

[61] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

[62] Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*, 2023.

[63] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

[64] Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. Lion: Adversarial distillation of closed-source large language model. *arXiv preprint arXiv:2305.12870*, 2023.

[65] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[66] Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*, 2023.

[67] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

[68] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[69] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[70] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

[71] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*, 2023.

[72] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

[73] nostalgebraist. interpreting gpt: the logit lens. `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`, 2020.

[74] Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

[75] Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. Jump to conclusions: Short-cutting transformers with linear transformations. *arXiv preprint arXiv:2303.09435*, 2023.

[76] Wes Gurnee and Max Tegmark. Language models represent space and time. *arXiv preprint arXiv:2310.02207*, 2023.

[77] Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. Neurons in large language models: Dead, n-gram, positional. *arXiv preprint arXiv:2309.04827*, 2023.

[78] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

[79] Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*, 2023.

[80] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[81] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[82] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

[83] Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

[84] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claim mentioned in the abstract is that we can efficiently finetune open-weight LLMs by combining layer pruning with quantization and low rank adapters, with minimal degradation performance on QA benchmarks. We provide empirical evidence for in §3. The scope of this work as claimed in the abstract is *open-weight LLMs*, and we show in §3 that our results hold across 7 different open-weight LLMs of varying size and model families.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In §4, we clearly outline the limitations of our empirical evidence – for example, the need for more comprehensive evaluations beyond QA benchmarks – and highlight future questions that our work did not cover.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Our work is primarily empirical and so we do not have any theorems.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We disclose all of the precise training details in §D.1 and all of the precise evaluation details in §D.2. Moreover, we ran all experiments with full determinism, such that all experimental results in our paper are fully reproducible, and we provide code snippets for fully deterministic training in §E.2. Most importantly, we describe our layer pruning algorithm in step-by-step detail in §2.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Unfortunately, we are unable to release the exact code we used to produce our results. However, all models, datasets, and training code are directly taken from open-sourced repositories (e.g. Hugging Face) and are noted as such in the text. With these models and data, the exact training and evaluation details are described in §D. Furthermore, as previously mentioned, all training runs were performed with full determinism, and we provide specific instructions instructions our setup in §E.2.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

Justification: For *training*, we provide all model configurations – Hugging Face model name, optimizer type, etc. – and hyperparameters decisions in §D.1. Note, we use the C4 dataset via Hugging Face (https://huggingface.co/datasets/c4), which does not have splits. For *evaluation* in §D.2, we provide complete instructions for reproducing our evaluation pipeline; in particular, we detail which datasets from Hugging Face to use, how we constructed n-shot prompts – including the split used to create the prompts – and which metric we compute.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In §E.2, we investigated the effect of changing finetuning seed on our results and report error bars. Note that error bars with respect to finetuning seed would be too small to be noticed if they were included on plots in the main paper.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss computer resources (GPU type, memory, total GPU hours for training) at the beginning of Appendix D. We also highlight the compute efficiency of our work at the end of our introduction (§1).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We do not conduct any research involving human subjects or participants, and all of the data used in this paper is publicly available and commonly used in the LLM literature.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of our work in Appendix F.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release any models or data, so the paper poses no such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all relevant models and datasets: for models, we provide Hugging Face download links and license information in the first table in Appendix D; we also give the download link and license information for C4, our finetuning dataset, in Appendix D; finally, we provide download links and license information for our evaluation datasets (the download links are given in bullets across §D.2, while the license information is given in footnote 4 at the beginning of §D.2). In all cases, the terms of use are properly respected.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release any new assets.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not run any crowdsourcing experiments with human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not conduct any studies involving human participants.

A Layer Pruning Strategy

A.1 Pictorial description of our layer pruning strategy

See Figure 4 below for a succinct description of our layer pruning strategy and empirical results.

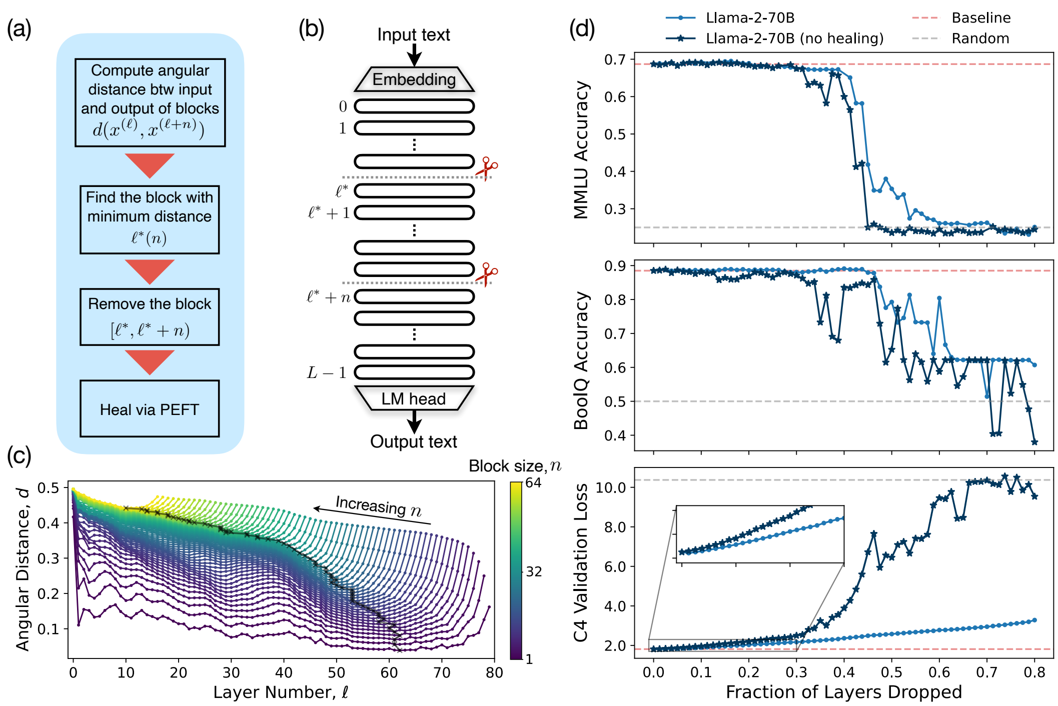

Figure 4: Overview of our layer-pruning strategy and example results: *(a)* a flowchart describing the algorithm: if removing n layers, we find the layer, ℓ^*, that minimizes the angular distance, d, between layers ℓ and $\ell+n$; we then remove the n layers beginning with layer ℓ^*; finally, if necessary, we can "heal" the damage with a small amount of (parameter-efficient) finetuning. *(b)* a schematic depicting the removal of n total layers, indexed from ℓ^* to ℓ^*+n-1. *(c)* angular distance, d, between different numbers of layers, n, vs. the layer number, ℓ, that indexes the beginning of the block of n; the bottom curve (darkest purple) represents $n=1$, while the top curve (lightest yellow) represents $n=64$; the black line traces $\ell^*(n)$, the minimum of the angular distance across the different sized layer blocks. *(d)* results of pruning Llama-2-70B with healing (light blue) and without healing (dark blue) as a function of the fraction of layers removed: the top (middle) panel gives the accuracy on the MMLU (BoolQ) question-answering benchmark, while the bottom panel the autoregressive loss on a subset of the C4 validation set; here, the dashed red lines (dashed gray lines) indicate the accuracy or loss of the original unpruned model (of random guessing); these plots illustrate that typical behavior we find in which there are sharp transitions in performance for the accuracy of question-answering tasks (here between 40%-50% pruning fraction), but continuity and very slow growth in the healed loss (dark blue) up to at least to 80% pruning fraction.

A.2 Comparing Simple vs. Similarity based pruning

In this subsection, we compare the similarity based pruning described in §2 to the simpler heuristic based pruning described in §3.3. We notice that before healing, the similarity-based method greatly outperforms the simple heuristic. After healing, we observe that the two methods perform comparably, but the similarity-based method marginally outperforms the simple heuristic.

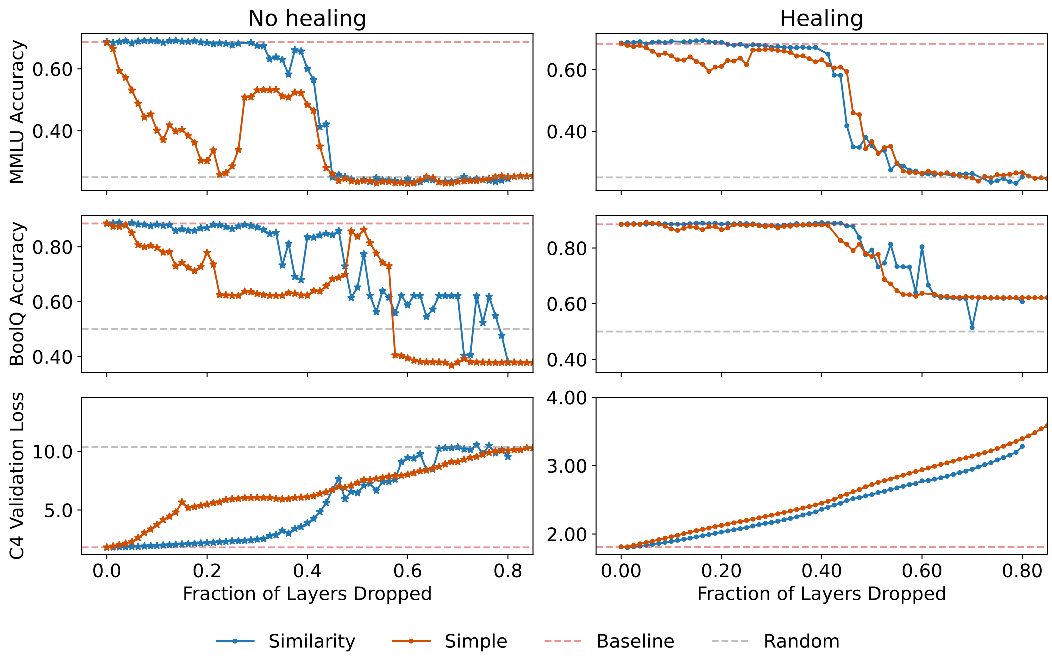

Figure 5: Evaluation of Llama-2-70B with the simple pruning heuristic (solid red line), shown along with scores for the similarity-informed pruning strategy (solid blue line), scores of the unpruned Llama-2-70B (red dashed line), and scores for randomly guessing (gray dashed line). (*Left:* before healing, *Right:* after healing; *Top:* MMLU, *Middle:* BoolQ, *Bottom:* C4 Validation Loss.) Without healing, the simple heuristic performs poorly across all evals; with healing, the scores of both methods are quite similar.

A.3 GSM-8K evaluation

In this section, we investigate the performance of our similarity-based metric described in §2 on GSM8K (8-shot) EM@1. We see in Figure 6 that performance immediately drops with layer pruning, unlike the results in §3; this suggests that deeper layers may be useful for harder reasoning tasks.

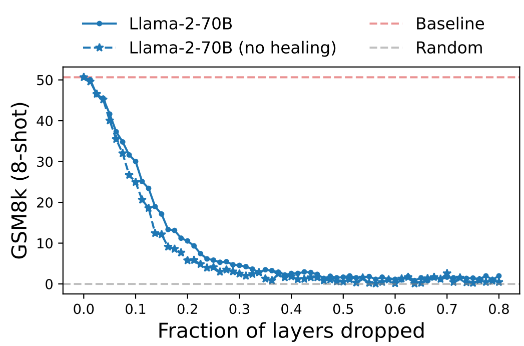

Figure 6: Evaluation of Llama-2-70B with the similarity-informed pruning strategy before healing (dashed blue line) and after healing (solid blue line) on GSM8k (8-shot). Unlike the question-answering benchmarks we studied in the main paper, we find an immediate degradation in performance, suggesting that deeper layers might be useful for harder reasoning tasks.

B Extended Literature Review

In this section, we review practical strategies for post-training efficiency and discuss some scientific investigations that provide motivation for, or insight into, our approach: in §B.1, we first review the history of pruning and then discuss its modern application to LLMs; in §B.2, we contrast pruning with distillation, an alternative strategy for reducing the parameter count of LLMs; then in §B.3, we discuss the various practical methods for efficient finetuning and inference acceleration that can be used in conjunction with our pruning strategy; finally in §B.4 we highlight some scientific investigations into some depth-dependent statistical properties of LLMs that are complementary to our results.

Note: As we were finalizing this work, a preprint of Ref. [22] was posted, which has a number of points of overlap with our work.

B.1 Pruning

Pruning is a method for reducing the size of a trained machine-learning model by removing unnecessary parameters, either individually or together as a group. Pruning for neural networks has a long history [23, 24], and, as originally conceived, *unstructured pruning* techniques sparsify networks by removing individual parameters based on pre-defined criteria. For instance, if a parameter of the model has a very small value, then removing it – i.e. by setting it to exactly zero – will likely have minimal impact on performance. Inspired by this early work, modern researchers began exploring different criteria for such unstructured pruning, focusing mostly on computer vision models [25–27]. In particular, Ref. [25] developed an *iterative pruning* method for alternatively pruning and finetuning a network in order to reach better compression ratios and performance.

While these models were smaller, they were not necessarily more efficient: sparsifying networks by removing individual parameters according to a criterion leads to irregular or pseudorandom sparsification patterns that are difficult to accelerate without specialized hardware or libraries designed for sparsity [28]. To that end, *structured pruning* techniques were developed to remove irrelevant groups of parameters together, such as particular channels or filters in convolutional networks. As this increased their practical relevance, researchers then began exploring structured pruning across computer vision [28–32] and pre-transformer NLP architectures [33–35].

Following unprecedented progress in language modeling, recent work has focused on applying structured pruning methods to the Transformer [36]. These studies consider nearly every possible component of the model architecture for elimination, with methods ranging from dropping attention heads [37–39], to dropping layers [40–45], to pruning hidden states [46], to rank reducing large weight matrices [47], replacing sparse weight matrices with smaller dense ones [48], to many combinations of the aforementioned groups [49, 50].

Of the prior work that also considers transformer layer dropping, most [40–42, 44, 49] study BERT-style models [51], while we consider decoder-only GPT-style models [1] that are most commonly used for large-scale language modeling and generation. BERT-style models are naturally suited for understanding tasks due to their bidirectional masked language modeling (MLM) objective, while GPT-style models are instead suited for generation, due to their autoregressive objective. While this divide has been questioned in light of more powerful GPT-style models [52], previous work [53] has found significant qualitative differences between BERT and GPT models in terms of the evolution of the layer-wise representation of words. Altogether, this suggests that layer-dropping strategies will behave differently between the two families.

One study for BERT-style pre-trained models, Ref. [44], concludes that the best layer-pruning strategy is dropping the final layers; this partially resonates with our results, although in contrast we find that *(a)* for some pruning sizes keeping the last few layers of the model is actually beneficial, and that *(b)* for all pruning sizes keeping the very last layer is essential. Additionally, while the authors also study similarity between representations in different layers – as in our approach – they actually found a higher similarity between representations in the shallow layers compared to the deeper ones – which very sharply disagrees with our results. Importantly, the models considered in Ref. [44] consist of a few hundred million parameters, which is much smaller than the model scales we consider in our work. Perhaps as a consequence, the authors didn't observe the sharp transition in downstream accuracies that we report in §3.1, despite the fact that they also finetuned their pruned models.

In contrast, while Ref. [43] does consider GPT-style models, the methodology is quite different from ours: *(i)* rather than pretraining first and then using a fixed layer-dropping strategy as we do, instead the authors incrementally drop layers in a modified pretraining procedure; and *(ii)* the authors study their own sub-1B parameter models, while we focus on the families of readily available, open-weight, large-scale 2.7B-70B parameter models that are commonly used and/or finetuned for practical applications.

Finally, a systematic approach to layer dropping in transformers has also been studied in the context of *wav2vec* models, which are encoder-only models that map speech to embeddings and are sized in the hundred-million parameter regime [54]. With these models, Ref. [45] developed a layer-pruning algorithm based on the correlation between layers and downstream metrics. Beyond the model architecture and domain, one significant difference between this and our work is that Ref. [45] considered non-contiguous pruning proposals, e.g. dropping alternate layers. Our intuition for layer pruning predicts that this shouldn't work as well – at least for decoder-only language models – as it creates multiple mismatches, one with each block of layers removed.

B.2 Model distillation

A completely different method for reducing the size of a trained machine-learning model is *model distillation* [55], in which knowledge is transferred from a large "teacher" model to a smaller "student" model by training the student on the distribution predicted by the teacher. The essential insight is that this can transform the very general knowledge and capabilities of the teacher into more streamlined, compressed, and possibly skill-specific representations.

While a very general technique, in the setting of language models, distillation has been implemented with *(a)* white-box approaches, in which the the student is trained to imitate the teacher's logits [56] or hidden states [57]; as well as with *(b)* black-box approaches, in which the student only has access to the output tokens generated by the teacher. This latter approach broadly covers cases where the student is trained on text that is augmented by the teacher in some way, such as by adding synthetic labels [58], generating high quality synthetic text [59–61] by providing chain of thought reasoning [62, 63], which aims to enhance the student's reasoning skills, or by annotating instructions that enhance the student's instruction-following capabilities [64].

Compared to layer pruning, these distillation methods require considerable computational resources due to the reliance on the large teacher to process a big corpus of data. Instead, our similarity-based pruning strategy only requires computing the similarity between representations at different layers on a small subset of a pretraining corpus, while our second simpler pruning strategy only uses the reduced model post pruning.

B.3 Efficient finetuning and inference acceleration

Complementary to directly reducing size of a model, *parameter-efficient finetuning* (PEFT) focuses on reducing the cost of specializing LLMs to certain tasks. In particular, Low Rank Adapters (LoRA) reduce the memory and compute of fine tuning by freezing the pretrained model and introducing a parametrically small number of additional trainable weights [65]. We use its quantized cousin, QLoRA [18], to keep our experiments cost efficient. Other PEFT methods that can be combined with our work are Refs. [66] and [67]: in the first, the initialization of the LoRA matrices is adjusted to a quantization scheme; in the second, LoRA ranks for different LLM modules are chosen in an adaptive manner.

For additional efficiency gains we could combine our layer-pruned models with methods that further accelerate inference: with speculative decoding [68], tokens are rapidly generated from a smaller draft model and then evaluated in parallel by the main model; with Medusa [69] the draft model is discarded for extra decoding heads, but ultimately achieves a similar effect. In particular, it could be interesting to consider highly-compressed layer-pruned models as potential draft models in a speculative decoding setup.

B.4 A breadth of depth-dependent studies

Finally, let us highlight some scientific work that study the depth-dependent properties of LLMs. One relevant direction considers how knowledge and linguistic properties are encoded in language models. On the one hand, Refs. [12, 70] analyze the *storage and recall* of factual associations: these works emphasize that knowledge localizes within the middle [12] or final [70] layers, which has implications for directly editing or erasing part of a model's factual knowledge. On the other hand, attempts to perform such editing gives evidence that information may be stored non-locally across layers [71]. Relatedly, Ref. [72] investigates the way facts are *processed* during inference, distinguishing between the role of attention heads, for attribute extraction, and the MLP blocks, for subject enrichment: both are delocalized across several layers.

Next, following the earlier "logic lens" [73], Ref. [74] invented a technique they called "tuned lens" to study the *trajectory of predictions* by using a learnable affine transformation to convert intermediate representations into a distributions over tokens (see also [75]). By studying the layer-to-layer dynamics of this distribution, the authors noted that it tended to converge. This convergence is very suggestive that that the deeper layers could be prunable, while the fact that they had to train an affine probe is likely related to our observation that the final layer cannot be pruned. Somewhat relatedly, Ref. [76] observed that geographic features in the underlying text can be determined from linear probes trained on intermediate activations, as long as the activations are deeper than halfway.

More abstractly, Refs. [77, 78] found that the sparsity of activations transitions at around halfway through a network's forward pass, evolving from sparse to dense. Perhaps relatedly, Ref. [79] investigated which model weights update the most during finetuning, finding that it's those in the mid-layers.

Altogether, these deep studies are complementary to our work, which, on the one hand, provides evidence that removing the deepest layers of an LLM does not significantly alter the model's performance, and, on the other hand, demonstrates a sharp pruning transition after removing approximately half of an LLM's deepest layers.

C Layer Pruning Intuition

Our intuition for layer dropping comes from thinking about the representations as a slowly changing function of layer index. In particular, the layer-to-layer evolution of representations for a transformer is given by a *residual* iteration equation

$$x^{(\ell+1)} = x^{(\ell)} + f(x^{(\ell)}, \theta^{(\ell)})\,, \tag{3}$$

where $(x^{(\ell)}, \theta^{(\ell)})$, respectively, are the multi-dimensional input and parameter vectors for layer ℓ, and $f(x, \theta)$ describes the transformation of one multi-head self-attention *and* MLP layer block. As for any residual network, if we unroll this iteration, we see that after L total layers the output is described as a sum over the transformations of all the layers

$$x^{(L)} = x^{(0)} + \sum_{\ell=0}^{L-1} f(x^{(\ell)}, \theta^{(\ell)})\,. \tag{4}$$

If the terms in the sum were *numerous*, ($L \gg 1$), and *independent*, e.g. if the block functions were instead a function of the overall input as $f(x^{(0)}, \theta^{(\ell)})$, then perhaps any particular contribution to the sum (4) could be neglected.

Of course, they are not at all independent: if we delete layer $\ell - 1$, then we must now connect the old input to that layer, $x^{(\ell-1)}$, into the block function of layer ℓ as

$$x^{(\ell+1)} = x^{(\ell-1)} + f(x^{(\ell-1)}, \theta^{(\ell)})\,, \tag{5}$$

where, for clarity, we are not relabeling layers or inputs despite the deletion. In general, such a *mismatch* between the original input and new input should be very damaging for the network. However, if, after some number of initial layers, the representations converge to a slowly changing function with respect to layer index,

$$x^{(\ell)} \approx x^{(\ell-1)} + \epsilon\,, \tag{6}$$

with $\epsilon \ll x^{(\ell)}$ in some appropriate sense, then the effect of deleting a particular layer ℓ, e.g. making the replacement $x^{(\ell)} \to x^{(\ell-1)}$ in going from (3) to (5), should only change the representation in the subsequent layer, $x^{(\ell+1)}$, by a small amount. Similarly, to successfully prune the n layers before layer ℓ, i.e. those indexed from $\ell - n, \ldots, \ell - 1$, we'd want that the input to the pruned block should be very similar to the output of the pruned block:

$$x^{(\ell)} \approx x^{(\ell-n)} + \epsilon \,. \tag{7}$$

Regardless, any layer removal has a cascading effect: since post pruning $x^{(\ell+1)}$ is computed by a different function than before, cf. (3) vs. (5), and since then $x^{(\ell+1)}$ is directly or indirectly input to subsequent layers, $\ell + 2, \ldots, L$, deleting a shallow layer should have a much greater impact than deleting a deeper layer.

From this, we have the following hypotheses that we will test experimentally:

(0) We should be able to prune layers of a residual network.

(1) We should have greater success pruning deeper layers.

(2) Blocks of layers we successfully prune should have outputs that are similar to their inputs.

In the next subsection, §2 we will explain the details of our pruning algorithm and in the following section, §3, we will present experimental evidence for points *(0)-(2)*.

D Experimental Details

Here we explain various details of models and healing (§D.1) and of evaluations (§D.2). *Note:* each model was trained and evaluated on a single A100 80GB GPUs, and no model's training required greater than 72 GPU hours.

D.1 Model and healing details

All models in this paper were fine-tuned using the Hugging Face `Trainer API` [80]. A list of models, their paths on Hugging Face, and their respective licenses are as follows:

Model	Repository Path	License
Llama-2 7B	`meta-llama/Llama-2-7b-hf`	Llama 2 Community License Agreement
Llama-2 13B	`meta-llama/Llama-2-13b-hf`	Llama 2 Community License Agreement
Llama-2 70B	`meta-llama/Llama-2-70b-hf`	Llama 2 Community License Agreement
Mistral 7B	`mistralai/Mistral-7B-v0.1`	Apache 2.0 License
Phi-2 (2.7B)	`microsoft/phi-2`	MIT License
Qwen 7B	`Qwen/Qwen-7B`	Tongyi Qianwen License Agreement
Qwen 14B	`Qwen/Qwen-14B`	Tongyi Qianwen License Agreement

For healing, we used the version of the Colossal Clean Crawled Corpus (C4) [81] from Hugging Face: `data = load_dataset("c4", 'en')`.[2] We truncated long examples as described later in the paragraph and added special tokens when available.[3] Models were finetuned for 5000 steps with a global batch size of 16: this corresponds to total finetuning tokens of $16 \times 5000 \times [\texttt{max\_seq\_length}]$ for each model. We used a cosine-annealed learning rate schedule, with a warmup of 100 steps. When possible, the peak learning rate was set to the peak learning rate from the model's pretraining; in practice, this means all models were trained with a peak LR of 3e-4, with the exceptions of Phi-2 [17], which was trained with a peak LR of 2e-4 during pre-training, Llama-2-70B, which was trained with a peak LR of 3e-5 (a value that resulted from a sweep), and Mistral-7B which was trained with a peak LR of 3e-6 (also a value that resulted from a sweep). All models 7B parameters or smaller were trained with a max sequence length of 2048 tokens, while all models 13B parameters or greater were trained with a max sequence length of 4096 tokens. While we realize that some models may have been pretrained on longer sequences, e.g. Qwen-*the-outlier* [15], we decided to the max sequence length consistent across models of similar size to allow fairer comparisons across model families.

On top of the Hugging Face Trainer API, we used quantization and Low-Rank Adapters (LoRA) [65] for all of our finetuning:

[2]This dataset is released with an Open Data Commons Attribution License (ODC-By).

[3]N.B. the Qwen tokenizer from Hugging Face does not include any special tokens; in this case, it was essential to add a default padding token.

- For quantization, we used the `bitsandbytes` library for QLoRA [18] to quantize our models to 4 bits.
- For LoRA, we used the Hugging Face `peft` library [82]. We set the LoRA dropout to 0.05 and kept the LoRA α equivalent to the LoRA rank, following [83]. Aside from two exceptions, discussed below, models are trained with LoRA rank 64.
- Also following Ref. [83], we only applied LoRA to FFN modules: `["gate_proj", "down_proj", "up_proj"]` for Llama-2 and Mistral models, `["fc1", "fc2"]` for Phi-2, and `["w1", "w2", "c_proj"]` for Qwen models.

The large majority of these hyperparameter choices are standard and found in previous works, e.g. Refs. [83, 84]. For absolute clarity, we list display all the model specific architecture and healing details below:

Model	# Layers	Vocab Size	Max Seq. Len.	FT Tokens	Peak LR	LoRA Rank
Llama-2 7B	32	32,000	2048	164M	3e-4	2
Llama-2 13B	40	32,000	4096	328M	3e-4	64
Llama-2 70B	80	32,000	4096	328M	3e-5	8
Qwen 7B	32	151,936	2048	164M	3e-4	64
Qwen 14B	40	151,936	4096	328M	3e-4	64
Mistral 7B	32	32,000	2048	164M	3e-6	4
Phi-2 2.7B	32	51,200	2048	164M	2e-4	64

We also have the following hyperparameters common between all models:

Config	Value
Finetuning dataset	C4
Batch size	16
LoRA α	LoRA rank
LoRA dropout	0.05
LoRA targets	FFN modules
LR scheduler	Cosine
Warmup steps	100
Total steps	5000

D.2 Evaluation details

We performed three principal evaluations: accuracy on *MMLU*, accuracy on *BoolQ*, and loss on *C4*.[4]

For **MMLU accuracy**:

- We use the `cais/mmlu` version of the dataset from Hugging Face.
- We follow the formatting suggested in the original reference [20] without further prompt engineering.
- For constructing few-shot examples, we use the `dev` set from `cais/mmlu`.
- For our experiments, we use 0 few-shot examples; our results and analysis are robust to this choice, cf. Figure 8.
- We report average accuracy across all subjects.

For **BoolQ accuracy**:

- We used the `hassansh/boolq_n_shot` version from Hugging Face.
- For our experiments, we use 0 few-shot examples.

[4]MMLU and BoolQ are released with an MIT license, while C4 is provided with an ODC-By license.

- The complete BoolQ results – truncated from the main text – are shown here in Figure 7: in the left panel we present the Llama-2 family, in the middle panel we present models from the Qwen family, and in the right panel we should Mistral-7B and Phi-2; we also make the experiments without healing semi-transparent in order to better display the results from the complete similarity-informed pruning method. Importantly, while we see here that healing plays a more important role than it did for MMLU in Figure 1, after healing we still have a characteristic flat region of robust performance; as before, the capabilities required to achieve a model's top score isn't removed by significant layer pruning until a critical model-dependent threshold.

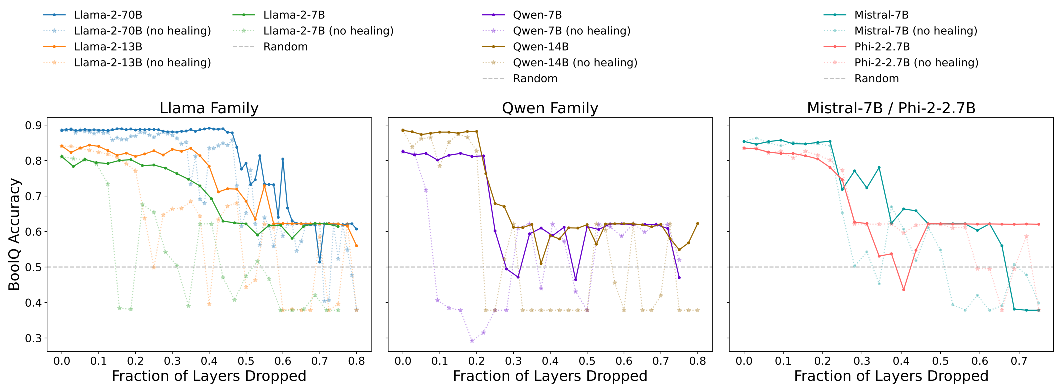

Figure 7: BoolQ accuracy (0-shot) vs. fraction of layers dropped for different model families. (*Left:* Llama-2 family; *Middle:* Qwen family; *Right:* Mistral-7B and Phi-2.) The solid lines represent performance after dropping layers and healing, and the (semi-transparent) dotted lines show performance after dropping layers only (no healing), and the dashed gray line is the score for guessing randomly. For BoolQ, healing leads to important improvements such that performances; then, across all models, performances are quite robust until 20%-55% pruning fractions, depending on model family and size, at which point they transitions to random guessing.

For **C4 Validation Loss**:

- We used the `c4` version from Hugging Face (soon be deprecated in favor of `allenai/c4`).
- We evaluated using the *validation* split as we healed with the train split.
- Given its size, we randomly sampled 60k sequences and held them fixed across all models.
- In Figure 2 we normalized the loss to facilitate fair comparison across model families that employ different vocab sizes: to normalize, we divided by $\log V$, where V is the *per-model* vocab size (listed in a table in §D.1). This, $\log V$, corresponds to the loss of sampling tokens uniformly, which naturally sets the scale for a given model.

E Ablations

Here we detail ablations of various hyperparameters: prompting (§E.1), finetuning seed (§E.2), LoRA rank (§E.3). Qualitatively, the results of the paper are quite robust to the variation of any of these.

E.1 Prompting

It's common knowledge that altering the prompt on QA evaluations can significantly impact results. To control for prompting, we ablate the MMLU accuracy for our principal similarity-informed pruning described in §2 when applied to Llama-2-13B: in the left panel of Figure 8, we show results for changing the ordering of the few-shot examples in the prompt, and in the right panel the same figure, we show results for changing the number of few-shot examples. Broadly we see that the layer-pruning method is robust to these changes.

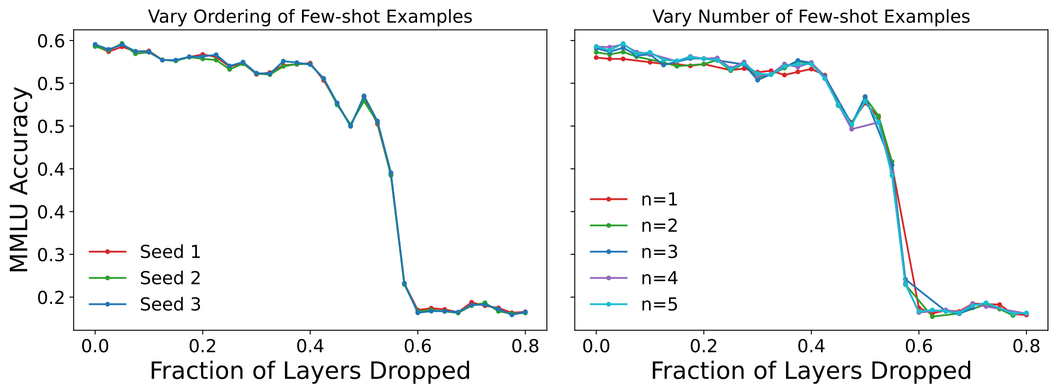

Figure 8: Effect of prompt ablations on MMLU accuracy vs. fraction of layers dropped for Llama-2-13B. *Left:* We vary the ordering of the few-shot examples and see it does not have any impact. *Right:* We very the number n of few-shot examples; while careful study of the flat region suggests increasing the number of few-shot examples marginally improves performance, regardless, the layer-pruning strategy is robust to this kind of variation.

E.2 Finetuning seed

Here we vary the finetuning seed. For all of our experiments, we use the following code snippet to ensure reproducibility:

```
SEED_VAL = 0
transformers.enable_full_determinism(SEED_VAL)
```

Since we begin with a pretrained model, the finetuning seed doesn't affect initialization, but it will impact the stochastic aspects of further training such as data order. To control for this, we ablate the finetuning seed for our principal similarity-informed pruning described in §2 when applied to Llama-2-13B: in Figure 9 we observe that the layer-pruning method is robust to the choice of seed.

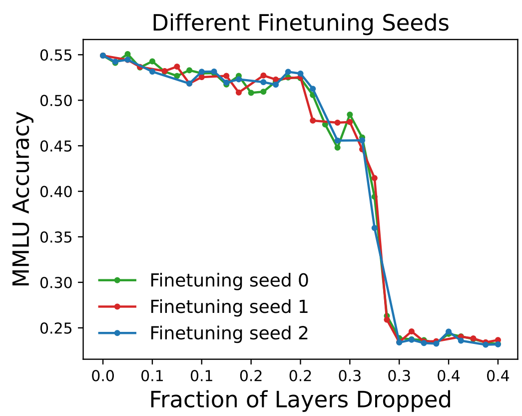

Figure 9: Effect of varying the finetuning seed on MMLU accuracy vs. fraction of layers dropped for Llama-2-13B: there is no meaningful effect.

E.3 LoRA rank

Here we vary the LoRA rank used for healing. Unfortunately, our compute budget did not allow us to make an exhaustive sweep across all of our experimental configurations. In lieu of that, we employed the following protocol for our main experiments:

- Begin with rank 64, following the QLoRA setup (see, e.g. Appendix B.2 of Ref. [18]).

- If healing with that rank significantly harms the performance compared to no healing, then sweep LoRA ranks for that model and, for the other evaluations, pick the best performing LoRA rank according to its MMLU accuracy.

This protocol is designed to maximize the chance that healing will improve performance across all of our evaluations. For simplicity, we ran this rank-picking protocol using the simple pruning heuristic, with the exception of Llama-2-70B.

In practice, this led to us using rank 64 for every model with the exceptions of Mistral-7B, with rank 4, Llama-2-7B, with rank 2, and Llama-2-70B, with rank 8. (To review this same information in tabular form, see the second Table in §D.1.) Figure 10 displays the sweeps over MMLU accuracy supporting these choices for Mistral-7B (bottom left panel), Llama-2-7B (bottom middle panel), and Llama-2-70B (top right panel): overall, while the LoRA rank does not have a significant impact on the qualitative behavior of the healed model, decreasing the LoRA rank generally improves performance. In the top left and middle panels of Figure 10, we show corresponding sweeps for Mistral-7B (top) and Llama-2-7B (middle) using the similarity-informed pruning strategy: we see that for this pruning method both models are much more robust, though rank 2 is still the top performing rank for Llama-2-7B.

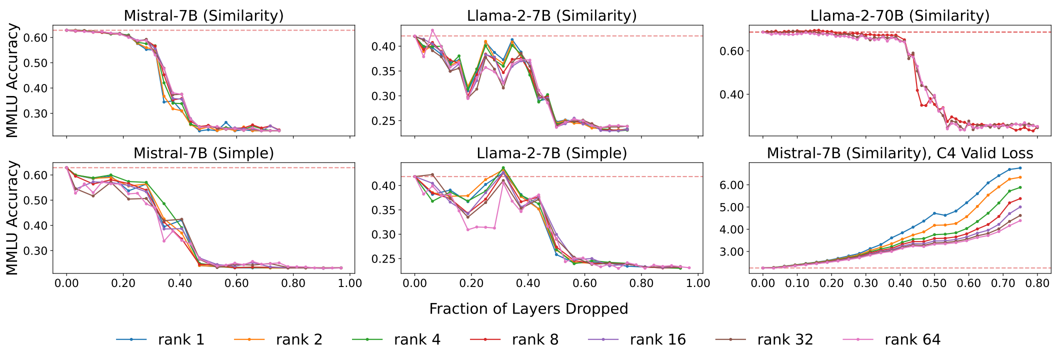

Figure 10: Effect of varying the LoRA rank. **Top**: 5-shot MMLU accuracy vs. fraction of layers dropped using the similarity-informed pruning strategy on Mistral-7B (*left*), Llama-2-7B (*middle*), and Llama-2-70B (*right*). Across all ranks we observe similar behavior, though there's a small effect of decreasing rank improving overall performance. **Bottom, left and middle**: 5-shot MMLU accuracy vs. fraction of layers dropped using the simple pruning heuristic on Mistral-7B (*left*) and Llama-2-7B (*middle*). As before, qualitative behavior is similar across ranks, though in this case it's much clearer that decreasing rank improves performance. **Bottom, right**: C4 validation loss vs. fraction of layers dropped using the similarity-informed pruning strategy on Mistral-7B. In contrast to MMLU, decreasing rank harms performance; together, these results suggest that larger ranks may be overfitting.

The characteristic improvement of MMLU accuracy with decreasing LoRA rank – even for extremely low ranks(!) – deserves an explanation. One possibility is that lowering the LoRA rank can better regularize finetuning against overfitting. In particular, astute readers may have been surprised at the discussion of peak learning rates in §D.1: models were finetuned with the same peak used in pretraining; a "large" LoRA rank of 64 introduces a number of additional parameters that may overfit to C4. This overfitting would certainly be harmful, since the actual pretraining datasets for the models we consider are *(a)* unknown to us, and *(b)*, likely to be of significantly higher quality than C4.

We investigate this directly for Mistral-7B. In the bottom right panel of Figure 10 we plot the C4 validation loss across different LoRA ranks: we see that while decreasing the LoRA rank generally improves MMLU accuracy (cf. left-most panels), at the same time it harms the C4 validation loss. This supports our overfitting hypothesis. In a greater-resourced future, it would be interesting to improve the healing process by considering other forms of regularization and learning rate tuning.

F Broader Impacts

This work studies methods for efficiently pruning open-weight LLMs. Positive societal impacts include an increased understanding of how LLMs process information across layers as well as the demonstration of potential practically useful techniques for improving the efficiency of LLM inference. Negative societal impacts are minimal; however, there may be possible second-order negative effects given that LLM systems are tools that can be used both positively and negatively, given different downstream use cases.