THE UNREASONABLE INEFFECTIVENESS OF THE DEEPER LAYERS

Anonymous authors
Paper under double-blind review

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 not useful for storing knowledge. To find these parameters, 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. In particular, we use parameter-efficient finetuning (PEFT) methods, specifically quantization and Low Rank Adapters (QLoRA), such that each of our experiments can be performed on a single A100 GPU. From a practical perspective, these results suggest that layer pruning methods can complement other PEFT strategies to further reduce computational resources of finetuning on the one hand, and can improve the memory and latency of inference on the other hand. 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 Radford et al. (2019) into useful products OpenAI (2022). As language model abilities improve OpenAI (2023); Gemini Team et al. (2023) 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 Gurnee et al. (2023); Zou et al. (2023); Clark (2019), or analyzing model activations Geva et al. (2020); Feng and Steinhardt (2023) and finding "circuits" responsible for certain behaviors Elhage et al. (2021); Wang et al. (2022). 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 Meng et al. (2022), subsequent work demonstrates that these methods and knowledge localization may be uncorrelated Hase et al. (2024).

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 sections of a network can be removed with minimal effect on its performance, then those parameters are likely not important for the specific task. Moreover, using model pruning as an intervention for understanding leads to practical results, as at the end of the investigation the smaller model performes better (or at least as well as the larger model) on the task at hand.

In this work we study a very simple pruning strategy using open-weight LLMs and measure performance degradation on common question-answering benchmarks. In particular, we develop a method that 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 result is that we can remove a substantial fraction of the *deepest layers* from models with minimal degradation in downstream

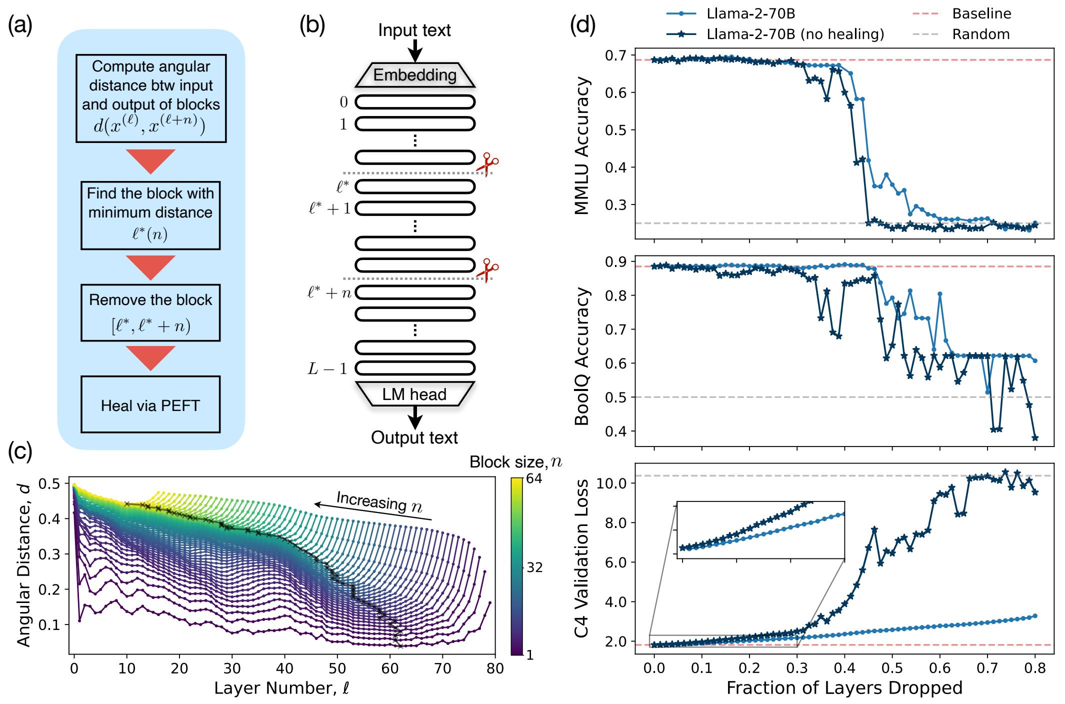

Figure 1: Overview of our layer-pruning strategy and example results: *(a)* a flowchart describing the algorithm: if removing n layers, we find the layer, ℓ^*, that minimizes the angular distance, d, between layers ℓ and $\ell+n$; we then remove the n layers beginning with layer ℓ^*; finally, if necessary, 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.

question-answering benchmarks. For example, for Llama-2-70B Touvron et al. (2023a) we can eliminate up to roughly *half* of the layers before the performance collapses on MMLU. An overview of our strategy and the results of pruning Llama-2-70B are shown in Figure 1.

In particular, our intuition for dropping layers comes from considering the residual structure of the transformer architecture. In more detail, the output of the final layer can be decomposed as a sum over the outputs of all the model layers plus the embedded input. If such a sum had numerous and independent terms, then removing a handful of them should not significantly change the output. However, since the terms are not independent – each layer is input to the following layer – we should expect to be able to remove terms if the residual contribution from a particular layer is small. In other words, if the output of each layer does not change too much from layer to layer.[1]

[1]This is strongly suggested by "lens" investigations that studied the evolution of the token distribution as a function of layer index such as the "logit lens" nostalgebraist (2020) and the "tuned lens" Belrose et al. (2023). A separate line of reasoning along these lines previously inspired neural ODEs Chen et al. (2018), and led

In conjunction with our layer pruning, we investigate the similarity of layer representations at different separations and find broadly that deeper layers are qualitatively more similar to neighboring layers than shallow layers (with the exception of the very final layer). This suggests an even simpler pruning strategy: remove layers beginning at the penultimate layer and proceed from deep to shallow until the desired number of layers have been removed.[2] In this case, we find that, after healing the damage with a small amount of QLoRA finetuning, we can achieve performance that nearly matches the more involved similarity-informed layer pruning strategy. The effectiveness of this method is evidence that LLMs might not properly leverage the parameters in the deeper layers of the network.

Overall, we hope you take these three bulleted points with you:

- The robustness of models to removing the deeper layers, the sharp transition in performance on downstream knowledge tasks (e.g. MMLU and BoolQ), and the smooth behavior of the autoregressive loss with respect to those pruning fractions, altogether suggest that the shallow layers may play a critical role in storing knowledge.
- The model's memory footprint *and* inference time decreases linearly with the number of removed layers.[3] This makes layer pruning a powerful tool, especially if the model's performance is robust to dropping layers.
- All the efficiency methods – pruning, PEFT and quantization – can be effectively combined with each other. Thus, in this work *each experiment was performed on a single A100 GPU* and is easily accessible to the open source and academic communities.

The structure of this paper is as follows. In §2, we first perform a literature review of both practical post-training strategies and science-of-deep-learning investigations that motivate our work. Then, in §3, we give intuition for our layer pruning strategy and explain our method in detail, while in §4 we iterate over all our experimental results. Finally, we conclude in §5 by highlighting directions of future work. Specific model, finetuning, dataset, and evaluation details can be found in Appendix B, and evaluations ablations can be found in Appendix C.

2 Literature Review

Pruning for neural networks has a long history (LeCun et al., 1989; Hassibi and Stork, 1992): while initial work focused on *unstructured pruning* (Han et al., 2015; Chen et al., 2015; Srinivas and Babu, 2015), *structured pruning* techniques were developed to make sparse networks more efficient (Li et al., 2016; Wen et al., 2016; Hu et al., 2016; He et al., 2017; Huang et al., 2018; Murray and Chiang, 2015; See et al., 2016; Kim and Rush, 2016). Recent work, of course, focused on structured pruning of transformers (Voita et al., 2019; Michel et al., 2019; Kim and Awadalla, 2020; Fan et al., 2019; Zhang and He, 2020; Fan et al., 2021; Jha et al., 2023; Sajjad et al., 2023; Liu et al., 2023a; Hou et al., 2020; Sharma et al., 2023; Ashkboos et al., 2024; Xia et al., 2022; Lagunas et al., 2021). Our work focuses on pruning the layers of decoder-only GPT style open-weight *large* language models after they've been pretrained. For an extended literature review, please see Appendix A.

3 Method

In this section, we give intuition for our layer pruning method (§3.1) and then we explain our method in detail (§3.2).

3.1 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

Ref. Yang et al. (2023) to argue that ideally representation should change substantially from layer to layer in order to most effectively make use of the parameters of a network.

[2]This strategy is especially interesting in situations where resource constraints inhibit the full application of the similarity-informed pruning algorithm described in Figure 2(a).

[3]Contrast this with quantization: the memory footprint decreases with the quantization ratio, but the inference time remains approximately fixed since parameters are typically de-quantized before any FLOPs.

is given by a *residual* iteration equation

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

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

$$x^{(L)} = x^{(0)} + \sum_{\ell=0}^{L-1} f(x^{(\ell)}, \theta^{(\ell)}) \,. \tag{2}$$

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 (2) 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

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

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,

$$x^{(\ell)} \approx x^{(\ell-1)} + \epsilon \,, \tag{4}$$

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 (1) to (3), 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:

$$x^{(\ell)} \approx x^{(\ell-n)} + \epsilon \,. \tag{5}$$

Regardless, any layer removal has a cascading effect: since post pruning $x^{(\ell+1)}$ is computed by a different function than before, cf. (1) vs. (3), 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, §3.2 we will explain the details of our pruning algorithm and in the following section, §4, we will present experimental evidence for points *(0)-(2)*.

3.2 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. (7) 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:

$$\ell^\star(n) \equiv \arg\min_{\ell} \; d(x^{(\ell)}, x^{(\ell+n)}) \,. \tag{6}$$

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]
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.

If fewer words inside of a figure are more helpful to you than the text in an enumerated list, then note that this algorithm is also depicted in panels (a)-(b) of Figure 1.

Elaborating on the first step, the angular distance on a single sequence of length T is given by

$$d(x^{(\ell)}, x^{(\ell+n)}) \equiv \frac{1}{\pi} \arccos\left(\frac{x_T^{(\ell)} \cdot x_T^{(\ell+n)}}{\left\|x_T^{(\ell)}\right\| \left\|x_T^{(\ell+n)}\right\|}\right), \tag{7}$$

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.[5] This distance should then be summed over a number of examples that is large enough to get a low-fluctuation estimate but overall should be quite small.

Elaborating on the “optionality” of the final step, we find that the near-lack of performance degradation on question-answering benchmarks, cf. Figure 1(d) and others in §4.1, can be extended to greater pruning fractions with a small amount of finetuning. Depending on resource constraints and intended application of the pruned model, this may not be necessary. However, the healing procedure does have a substantial impact on perplexity, cf. Figure 1(d) and others in §4.2.

For both the angular distance measuring and the healing, if the ultimate goal is to supervise finetune (SFT) a model for a downstream task, it could be useful to evaluate the distance of a sample from that dataset and then combine the healing process with the SFT. In contrast, for the greatest generality, it’s most natural to measure distance and heal with a pretraining dataset that approximates the statistics under which the model was originally pretrained.

Finally, we also investigated an even simpler pruning strategy inspired by analyzing the angular distances across different model families: drop the deepest layers, excluding the final layer before the LLM head, and then (*non-optionally*) heal the damage. For complete clarity, this means that if we are pruning n layers from an L-layer model, then we would remove layers $(L - n)$ to $(L - 1)$, inclusive.

4 Results

In this section, we demonstrate the effectiveness of our pruning strategy on different question-answering (QA) benchmarks and highlight a robust pruning-driven transition in performance (§4.1), while, in contrast, we find that the autoregressive perplexities of the healed pruned models are continuous across their transition points (§4.2); then, after comparing the similarity statistics between different layers across model sizes and families (§4.3), we contrast our principal similarity-informed pruning strategy with a simpler remove-the-deepest-layers strategy (§4.4).

For our experiments, we pruned 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 Touvron et al. (2023a), the Qwen family Bai et al. (2023), Mistral-7B Jiang et al. (2023a), and Phi-2 Javaheripi and Bubeck (2023). For these models, we executed the “healing” step using QLoRA Dettmers et al. (2023): 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) Raffel et al. (2020), a common pretraining dataset. As a result, *each experiment of ours was performed on a single* A100 *GPU*. For our QA evals, we used Massive Multitask Language Understanding (MMLU) Hendrycks et al. (2020), a common world-knowledge and problem solving benchmark, and BoolQ Clark et al. (2019), a common yes/no reading comprehension benchmark where the answer has to be inferred from the text itself. The specifics of our models, healing procedure, dataset choices, and

[4]Layers are often contained in a data structure, such a `ModuleList` in *PyTorch*, so to drop these layers we would simply define a new `ModuleList` that removes the layers from $\ell^\star$ to $\ell^\star + n - 1$.

[5]Two comments: *(i)*, we do not expect our choice of angular distance – in lieu of any other reasonable metric, e.g., such as cosine similarity – to be particular significant; and *(ii)*, we chose to focus on the final token since, due to the causal attention mask, its embedding is the only one that depends on the entire sequence.

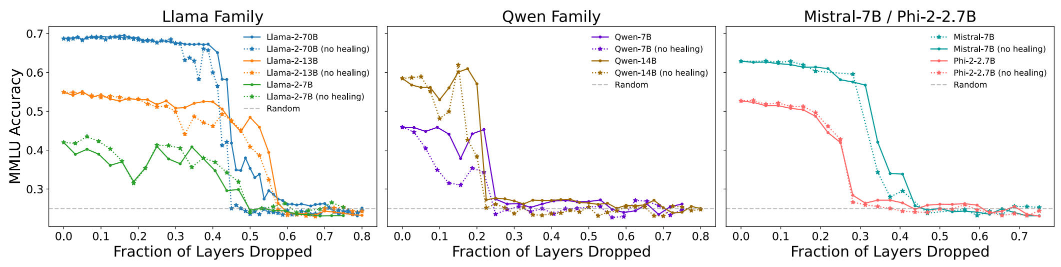

Figure 2: MMLU accuracy (5-shot) vs. fraction of layers dropped for different model families. (*Left:* Llama-2 family; *Middle:* Qwen family; *Right:* Mistral-7B and Phi-2.) The solid lines represent performance after dropping layers and healing, dotted lines show performance after dropping layers only (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.

evaluation details can be found across Appendix B; ablations of different hyperparameter choices can be found across Appendix C.

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

Our first set of results are shown in Figure 2, where we plot 5-shot MMLU accuracy as a function of the fraction of layers removed: 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 show Mistral-7B and Phi-2. In order to better compare models of different total number of layers, in these plots we opted to normalize the x-axis by the fraction of layers removed (rather than the absolute number of layers removed). Note that since MMLU contains multiple choice questions with four possible responses, the expected accuracy of random guessing is 25%.

Importantly, we see 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.[6] 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 §4.3.

4.2 Analyzing loss on next-token predictions

In this section, we look at the effect of layer pruning on the pretraining optimization objective – the cross-entropy loss of next-token prediction – when evaluated on a subset of the C4 validation dataset.[7] In order to have a fair comparison across models with different sized vocabularies V, we normalize the loss by $\log V$, which corresponds to the loss of sampling tokens randomly with uniform probability. (See Appendix B.2 for more details.)

[6]This effect is rather robust to choice of QA benchmark: in Appendix Figure 7 we plot the average 0-shot BoolQ accuracy for our model families and observe analogous behavior.

[7]We make sure that none of the validation data are seen during the healing stage.

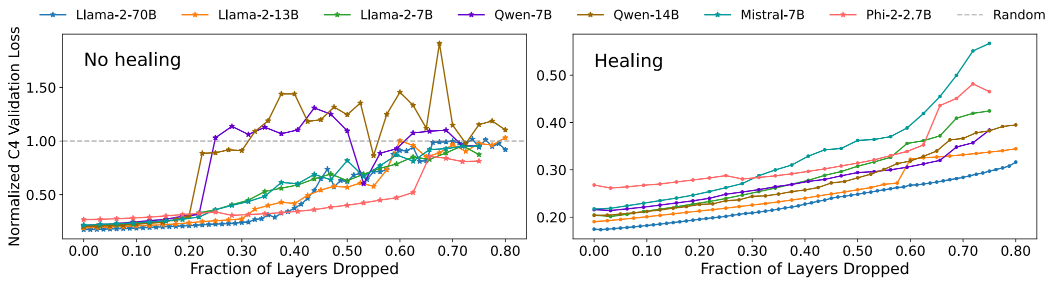

Figure 3: Normalized C4 validation loss vs. fraction of layers dropped before healing (*left*) and after healing (*right*); each curve is normalized by the cross-entropy loss of sampling uniformly from the model's vocabulary. For the experiments before healing, the loss for each model transitions to 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 2. Contrasting the overall scale of both plots, it's clear that healing significantly restores the performance on next-token prediction to near-unpruned levels.

In Figure 3 , we plot the normalized C4 validation loss for all seven of our models, after healing (left panel) and before healing (right panel), as a function of the fraction layers removed. Without healing, we see that there is a somewhat sharp(ish) transition to random guessing for each model at approximately the pruning fraction that the QA benchmark accuracies also sharply transition to random guessing, suggesting that models are hopelessly harmed at this point, cf. Figure 2. Next, 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. Most strikingly – from a scientific perspective – is the post-healing continuity through the pruning fractions where we previously found sharp transitions for the QA benchmarks: this decoupling illustrates one way of disconnecting (or creating a miscalibration) between performance on downstream tasks – such as MMLU and BoolQ – and continuous measures of performance – such as the cross-entropy loss. [8]

Overall, the slow linear increase in cross-entropy loss suggests that deeper layers may be used for some other ability that is learned during pre-training. In Section 4.5, we evaluate pruned models on a wider suite of tasks and find that that one of these abilities may be higher-level reasoning.

4.3 Angular distances between representations

Given the central role the angular distance (7) plays in our pruning strategy, let's take a subsection to look at 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.

Recall from earlier Figure 1(c): for Llama-2-70B this plotted the angular distance $d(x^{(\ell)}, x^{(\ell+n)})$ that compared the ℓ-th layer to the $(\ell + n)$-th layer, across all initial indexes ℓ for block sizes from $n = 1$ to $n = 64$; the minimum of the curves, $\ell^\star(n)$, gave the optimal block to prune for a given n, cf. (6).

A more compact way to display this same data is shown in the heat maps of Figure 4: 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 generalizations: *(i)* the smallest distances are found across the deeper blocks, meaning deeper layers are typically quite similar to each other and can be more easily dropped; *(ii)* the distances across the deepest blocks – the blocks that include the last layer – take either maximal or nearly-maximal values, meaning one should never drop the final layer. While

[8]This is consistent with Ref. Schaeffer et al. (2023) that argued jumps in one kind of metric may not be visible in others.

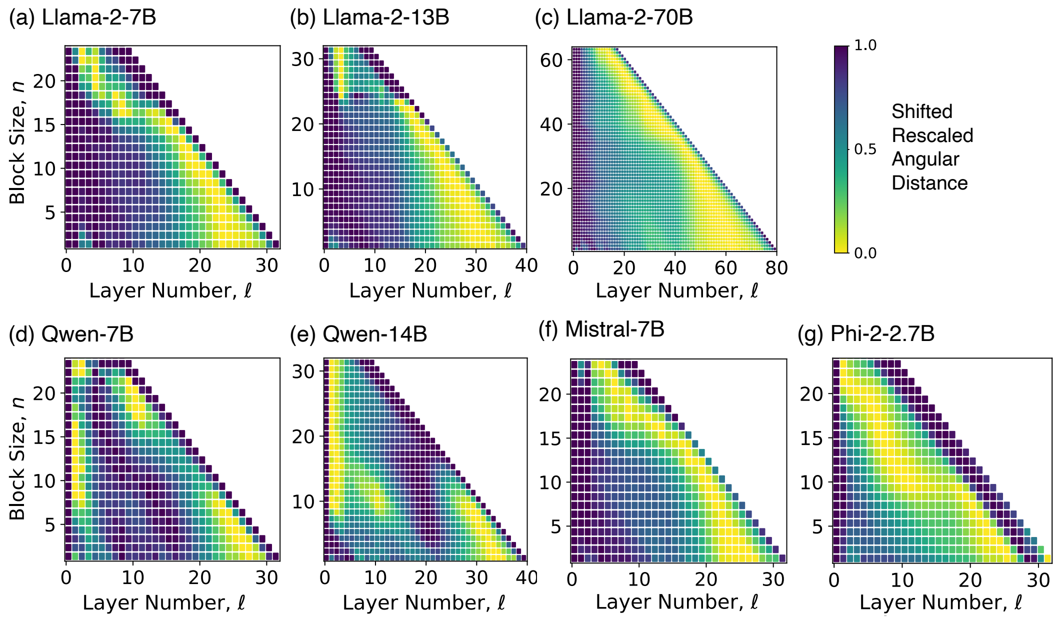

Figure 4: Normalized angular distance (7) from initial layer ℓ (x-axis) with block size n (y-axis) for each of the seven models we evaluated; the distance for each n is shifted and rescaled to span the same range, $[0, 1]$ (yellow to purple): the optimal block to prune, $\ell^*(n)$, corresponds to the 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.

broadly true, there are a few exceptions. For some models, e.g. Phi-2-2.7B, or for the largest blocks in some models, e.g. Llama-2-7B, final *few* layers seem important. As previously noted, the Qwen family is somewhat unusual: here we see that there are a few odd "islands" of high similarity for shallow blocks; this likely explains the shorter region of robust performance in Figure 2.

4.4 A SIMPLER PRUNING STRATEGY

Inspired by our recent conclusions, 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. Compared with our principal similarity-informed pruning strategy, this simpler heuristic algorithm has the advantage of never requiring practitioners to load onto a GPU or inference the unpruned model. It also provides a meaningful ablation of the importance of optimizing the block to prune.

In Figure 5, we contrast our two pruning strategies, both before healing (left panels) and after healing (right panels), for the QA benchmarks (MMLU/BoolQ, top/middle panels) and the autoregressive loss (C4 validation, bottom panels). On the one hand, the simple heuristic performs quite poorly without healing the damage incurred by pruning: accuracy on the QA benchmarks decays rapidly to (near-) random with increased pruning fraction, and the loss begins to increase very rapidly even with small amounts of pruning. On the other hand, the results for the two pruning strategies across evaluations are quite comparable after healing: for the QA benchmarks, the similarity-informed algorithm slightly better preserves the accuracy before the phase transition, though the simple algorithm perhaps pushes the phase transition to slightly greater pruning factions; and for the loss, the curves nearly lie on top of each other, though 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.

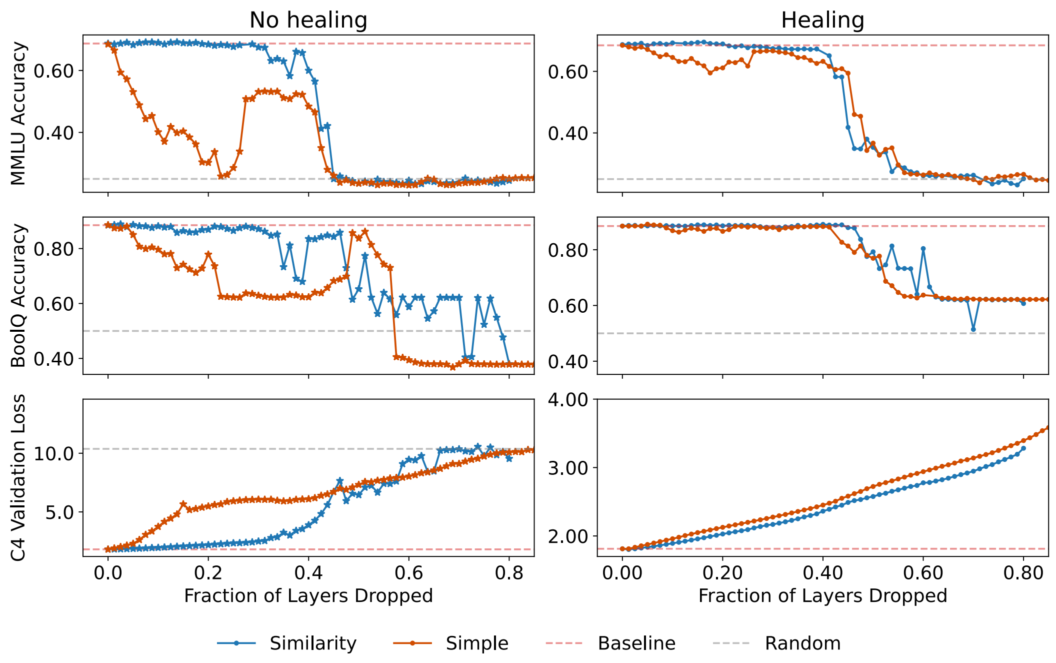

Figure 5: Evaluation of Llama-2-70B with the simple pruning heuristic (solid red line), shown along with scores for the similarity-informed pruning strategy (solid blue line), scores of the unpruned Llama-2-70B (red dashed line), and scores for randomly guessing (gray dashed line). (*Left:* before healing, *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.

4.5 What are deeper layers doing?

In previous sections, we provide evidence that deeper layers do not affect performance on MMLU (see Figure 2); however, the immediate degradation in performance in Figure 3 suggests that deeper layers are useful for other capabilities. What are deeper layers doing? In this section, we evaluate pruned models on a suite of different tasks to better understand what abilities model lose when pruning deeper layers. We choose one evaluation from common subcategories for language model evaluation[9], as well as an evaluation for Chain-of-Though-MMLU (CoT-MMLU) where evaluation is done allowing the model to produce chain-of-thought outputs. See Section C.5 for more details on the evaluation setups. We then perform layer dropping via our cosine-similarity cutting method and the simple baseline from Section 4.4, and evaluate performance. In Figure 6, we observe that changing the evaluation task can significantly change the performance of layer dropping. Interestingly, moving to generation-based tasks (summarization or CoT-MMLU) retains the same qualitative behavior (e.g. a relatively flat region of performance followed by a sharp dropoff around 45-55% pruning fraction). On the other hand, evaluations that require reasoning capabilities (GSM8k or HellaSwag) exhibit immediate degradation in performance. We observe similar trends when we evaluate models at larger scale (see Figure 12). Overall, this suggests that deeper layers are useful for higher-level reasoning tasks, but relatively less important for summarization and knowledge-intensive QA tasks.

5 Discussion and Future Directions

We leverage model pruning as a tool to understand how open-weight LLMs store knowledge, and show that we can prune a significant portion (up to 50%) of deeper layers with minimal impact on knowledge-QA performance. This suggests that shallow layers are important for storing knowledge.

[9]We use the subcategories from Section 2.3 of the Llama-2 Touvron et al. (2023b) paper

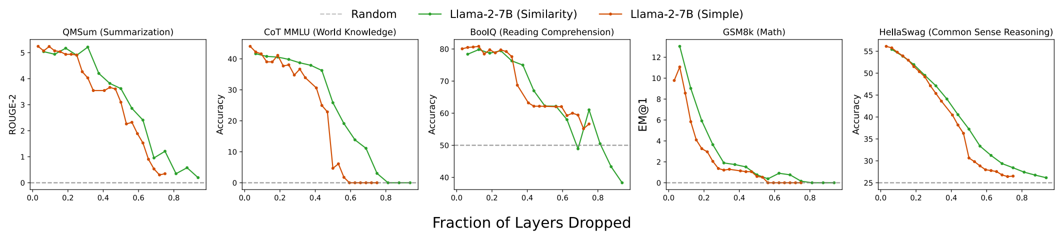

Figure 6: Evaluation of Llama-2 7B using the cosine-similarity cutting method (green) and the simple baseline from Section 4.4 (red) across different evaluation tasks. **Left to Right**: QMSum, CoT-MMLU, BoolQ, GSM8k, HellaSwag. We observe that reasoning tasks (GSM 8k and HellaSwag) show immediate performance degradation when pruning deeper layers.

While model pruning experiments may be a computationally expensive tool for scientific understanding of LLMs , we leverage advancements in efficient inference and fine-tuning to lower computational requirements: beginning with the release of the open-weight LLaMA family (Touvron et al., 2023b), the open-source machine-learning community has rallied around the philosophy of making LLMs accessible to everyone. This has engendered many innovations around efficiency, such as LoRA Hu et al. (2021) and quantization (with LoRA) Dettmers et al. (2023), allowing large (near-)state-of-the-art 70B models to be finetuned on only single 80GB A100 GPUs. As a result, all of our experiments are performed on a single A100 GPU and are easily accessible. We hope our work motivates the broader community to use model pruning as a tool for understanding where knowledge is stored in open-weight LLMs.

At the conclusion of the work, we are left with the following questions:

- What are better layer-pruning strategies? What are better approaches to healing?[10]
- 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?
- Relatedly, is knowledge generally stored in shallow or middle layers, or is it delocalized?
- Do pretraining details affect the ability to prune, e.g., are scaling-law over-trained or distilled models more difficult to prune?
- How can we enable LLMs to more effectively use the parameters in their deepest 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 prunable parameters? Others suggest explorations with different pretraining architectures and objectives, e.g. in order better make use of the deeper layers (for example, one can imagine applying layer dropout Fan et al. (2019) or early exit during pre-training Elhoushi et al. (2024) to induce equal usage of 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.[11] It would be very interesting to use pruning to systematically study these kind of interpretability questions.

[10]At the cost of introducing another hyperparameter and requiring both pruned and unpruned models to fit in memory during finetuning, one natural way to improve healing is by adding an auxiliary student-teacher loss that explicitly addresses the pruning mismatch (5), such as

$$\mathcal{L}_{\text{aux}} \sim \left(x^{(\ell^*+n)}(\theta_0) - x^{(\ell^*)}(\theta)\right)^2 , \tag{8}$$

where θ_0 are the frozen parameters of the unpruned model, and θ are the parameters of the pruned model to be healed; thus, $x^{(\ell^*+n)}(\theta_0)$ is the input to the $(\ell^* + n)$-th layer in the unpruned model, $x^{(\ell^*)}(\theta)$ is the input to that same layer after pruning, and $\mathcal{L}_{\text{aux}}$ minimizes their mismatch. We thank Sho Yaida for this observation.

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

REFERENCES

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.

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

OpenAI. Gpt-4 technical report, 2023.

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.

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.

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.

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

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

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

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.

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.

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.

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.

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*, 2023a.

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

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.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023.

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.

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.

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.

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.

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

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

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.

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.

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.

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.

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

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

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

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.

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

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

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

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.

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.

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.

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.

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

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.

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.

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.

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

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

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.

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*, 2023a.

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

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

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.

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.

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.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *arXiv preprint arXiv:2304.15004*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023b.

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.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

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.

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.

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.

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

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.

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.

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

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

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.

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.

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

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*, 2023a.

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.

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.

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.

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

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*, 2023b.

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.

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.

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.

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

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.

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.

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.

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

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

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, 2023b.

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

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.

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.

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.

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

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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*, 2021.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, et al. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*, 2022.