# The impact of LLM pruning for fine-tuning

Stanford CS224N Custom Project

**Varun Shanker**
Department of Biophysics
Stanford University
vrs@stanford.edu

**Sarah Chung**
Department of Computer Science
Stanford University
ischung@stanford.edu

## Abstract

Scaling laws in the deep learning era have inspired the scaling of large language models (LLMs) to parameter sizes that require extraordinary amounts of compute and energy for both training and inference. Such large pretrained models, or foundation models, provide valuable model initializations for subsequent fine-tuning. Recent work has converged upon the discovery that the deeper layers of many GPT-style decoder only LLMs are redundant. In this work, we explore the robustness and general applicability of LLM pruning to applications in non-traditional language modeling domains, like coding. We find that the identification of candidate layers for pruning is generally insensitive to the class of text being modeled. However, the performance of emergent capabilities quickly decays with the loss of these layers.

## 1 Key Information to include

- Mentor: Yann DuBois
- External Collaborators: None
- Sharing project: None

## 2 Introduction

Large language models (LLMs) trained over vast amounts of text and information from diverse sources have proven to be unreasonably effective across many settings and applications. In recent years, new fundamental insights into scaling laws governing neural language models coupled with recent empirical evidence have motivated the emerging trend of developing LLMs with increasing number of parameters (Kaplan et al., 2020; Hoffmann et al., 2022). Notably, since the development of the original Transformer model, the sizes of popular models using related architectures have been growing exponentially year-on-year Mauboussin (2023). While expanding model size has indeed been a successful solution to improve performance, these gains come at a substantial cost at many levels as they demand increased computing power for inference, memory to host, and energy consumption to train HAI (2024). As a result, there remains a critical need for rational methods that enable the development of compute and memory efficient LLMs.

An alternative to simply training smaller models is converting large language models into smaller. One such strategy involves pruning of components of the neural network that are identified to either be redundant or not having a direct impact on the output. A key motivation for pruning a larger model rather than directly training more parameter efficient models stems from the observation that scaling of large language models not only yields improved performance, but also gives rise to improved sampling on a wide range of tasks not directly modeled during pretraining, often considered to be emergent capabilities Brown et al. (2020). Pruning of redundant components of pretrained large language models, thus may not only benefit compute efficiency, but also preserve the emergent capabilities learned. Beyond general zero-shot performance on diverse tasks, in many cases performance on these emergent properties of LLMs can be substantially enhanced with supervised fine-tuning on curated datasets. This potential to fine-tune a single model for high accuracy across many domains is a highly attractive feature of pretrained LLMs. However, since compute requirements for fine-tuning of pretrained large language models for downstream tasks also grow linearly with the underlying model size, we anticipate that methods that can effectively eliminate the existing reliance of LLMs on expensive and broadly inaccessible, advanced hardware will be valuable. In this work, we explore whether pruning layers of LLMs may offer a practical and robust alternative to relying on compute-intensive larger LLMs model without compromising performance or versatility.

## 3 Related Work

Pruning of existing performant pretrained LLMs has emerged as one prominent strategy towards this goal of producing highly efficient models. Importantly, pruning is distinct from a related concept known as model distillation that focuses on reducing the parameter count of LLMs by teaching a smaller network to mirror a larger pretrained one (Hinton et al., 2015; Shleifer, 2021; Sanh et al., 2022). The concept of post-training pruning of neural networks to shed weights that have nominal impact on model predictions has been explored using many techniques. These methods vary in both the components of the networks they consider to eliminate and the criteria used to identify candidate elements to remove. Early efforts centered around validating the feasibility of this concept without incurring substantial "brain damage" utilized brute-force approaches to prune weights with the least impact on the loss function (LeCun et al., 1990; Hassibi and Stork, 1993). Since then, more sophisticated methods for identifying the least important or redundant weights have been developed to stay in-step with the growing sophistication of the underlying algorithms themselves. Modern post-training pruning techniques in the deep-learning era can be largely classified into two broad categories: i) structured and ii) unstructured pruning. The first uses techniques to produce sparse networks by removing individual parameters, whereas structured pruning considers the elimination of entire groups or sets of parameters together. Specifically, in contrast to the sparse networks produced by unstructured strategies, structured pruning is particularly well-suited for integration with methods for acceleration without requiring bespoke hardware or embedded systems capable of handling such sparsity Anwar et al. (2017).

Many new frameworks for structured pruning of the Transformer and Transformer-related models have been proposed recently that consider dropping parameters from nearly every component of the model architecture. Specifically, approaches have studied eliminating attention heads, layers, hidden states, rank reducing weight matrices, and sparse weight matrices (Fan et al., 2020; Brix et al., 2022; Zaheer et al., 2020; Michel et al., 2019). While in this work, we only consider GPT-style decoder only models that have become popular base models for conversation-handling LLMs due to their capabilities for language generation with auto-regressive conditioning, another prominent class of models used for language tasks is the BERT-style encoder models that have a bidirectional masked language modeling objective. Importantly, this fundamental difference in modeling between BERT and GPT models is appreciable when analyzing their layer-wise embeddings of tokens. It is thus unsurprising that pruning strategies, particularly layer-based, will not be immediately transferable across model classes. For example, when studying BERT-style pre-trained models, it was found that dropping the final layers is the most beneficial strategy although the greatest similarity in adjacent layers is found in the shallow layers(Devlin et al., 2018; Sajjad et al., 2023).

Most relevant to this study, several independent groups have contemporaneously discovered that GPT-style decoder-only networks have a surprisingly large fraction of layers that have a negligible role in network function (Russak, 2024; Gromov et al., 2024; Razzhigaev et al., 2024; Men et al., 2024; Samragh et al., 2023). One study particularly shows that, across base GPT-style LLMs ranging in sizes from 7 billion to 70 billion parameters, up to 40 % of the layers can be dropped until a functional cliff is reached and performance substantially degrades Gromov et al. (2024). However, this work was limited in scope. Claims that a significant block of layers were redundant were only substantiated with benchmarks against general knowledge and boolean question datasets. Motivated by this finding, in this study, we seek to i) explore the robustness of the layer pruning strategyii) evaluate the generalization of this finding to harder tasks and iii) determine whether fine-tuning of pruned LLMs can recover any losses in performance with respect to the original larger model.

## 4 Approach

Our approach includes a set of experiments that, taken together, further the current understanding of the functional value of the deeper layers in large language models. First, we consider the robustness of identifying unimportant layers by using diverse datasets to compute metrics of layer similarity across various block sizes. We then use these results to inform the ideal starting layer from which we prune a quarter of the total layers in a pretrained large language model. Next, we benchmark the performance of the pruned model against the original base model on coding, as a representative task to test the hypothesis that pruning may lead to some extent of "brain damage" on emergent abilities learned at scale. Finally, we perform a form of reinforcement learning on an supervised instruction-following version of a large language model to determine whether additional stages of learning leads to any changes in the contribution of the deeper layers and may benefit from not undergoing pruning prior to finetuning (Figure 1).

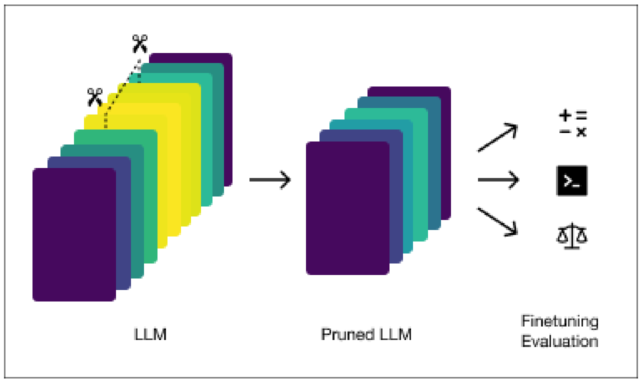


Figure 1: Conceptual framework for structured pruning of layers and downstream evaluation and fine-tuning of pruned LLMs

## 5 Experiments

### 5.1 Data

For this study, we utilize several well-known datasets to answer distinct questions. First, to determine whether metrics of layer similarity used as criteria to select layers for pruning are sensitive to the type of input, we used the c4, ultrachat, and Chinese Medical datasets (Gao et al., 2020; Chong et al., 2022; Meng et al., 2023). C4 is a colossal, cleaned version of Common Crawl's web corpus, and most notably used to train the T5 Transformer model. UltraChat is an open-source dialogue dataset that is commonly used to finetune pretrained LLMs for enhanced conversational capabilities. Finally, the Chinese Medical Dialogue dataset is a corpus of medical dialogue written in chinese characters. Importantly, Llama3 was trained over data comprised of over 30 languages, of which chinese mandarin is included. To analyze layer similarity, 100 samples from each dataset are used as input strictly for inference.

As the representative task for the emergent capabilities of LLMs, we utilize the OpenAI HumanEval dataset comprised of 164 coding problems and solutions written in python and contain comments in english. The HumanEval benchmark is most often used to assess the functional correctness of language model outputs,

For fine-tuining of Llama3-8B and the pruned corresponding version we use the Orpo-Dpo-Mix-40k that was curated using a broader set of preference datasets. The input is a given prompt, the chosen response, and a rejected response. This dataset can be used with ORPO for supervised fine-tuning

### 5.2 Evaluation method

We consider two primary metrics in this study. Central to the efforts analyzing the potential for pruning of redundant layers in large language models is the determination of similarity between layers. The key intuition underlying why deleting layers after pretraining may be an acceptable strategy is noted when considering the output of a given layer to be simply the sum of the input going into the layer and a residual, or perturbation made by the layer

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

where $(x^{(\ell)}, \theta^{(\ell)})$, respectively, are the multi-dimensional input and parameter vectors for layer $\ell$, and $f(x, \theta)$ describes the transformation of one multi-head self-attention *and* MLP layer block. If the output after some number of layers $L$ begins to converge such that

$$x^{(\ell)} \approx x^{(\ell-1)} + \epsilon\,, \tag{2}$$

for $\epsilon \ll x^{(\ell)}$, then deleting this layer may not have a substantial impact to the network Gromov et al. (2024).

To accomplish this, we utilize the angular distance of the final token between layers as the relevant representation that was also employed by motivating work. The angular distance, $d$ is defined as a rescaled cosine similarity between the hidden dimensions of the final token between two layers. For clarity, the angular distance between a pair of layers separated by a block size of $n$ for a given input of sequence length $T$ is given by:

$$d(x^{(i)}, x^{(i+n)}) \equiv \frac{1}{\pi} \arccos\left( \frac{x_T^{(i)} \cdot x_T^{(i+n)}}{\left\| x_T^{(i)} \right\| \left\| x_T^{(i+n)} \right\|} \right)\,, \tag{3}$$

To perform a comprehensive analysis of layer similarity for a given model, we use 100 samples from a dataset and average the angular distance using equation (1) between the starting and final layer. For visualization of similarity maps, we plot the normalized distance for such that each n is shifted and rescaled to span the same range, [0, 1] as was previously shown Gromov et al. (2024).

For analysis of coding knowledge on the HumanEval benchmark Chen et al. (2021), we consider the Pass@1 metric. We consider this the most relevant measure of assessing whether the model has incurred any "brain damage" after pruning and its practical impact on the end user.

## 5.3 Experimental details

In all experiments, we use the Llama3 8 Billion parameter decoder-only large language model (Llama3-8B) as the base pretrained model. To demonstrate the generality of our approach and potential for utilization in even low-resource settings, we dynamically quantize the model precision to 16 bits in all experiments for decrease memory footprint and allow all experiments to be conducted on either a single L4 or A100 GPU.

For evaluation on the HumanEval benchmark Chen et al. (2021), we generate one completion for each model tested and score the results to compute Pass@1 measure on the test dataset which ranges from 0 to 1. We set the max legnth of the output to be 512.

To assess the impact of further finetuning or reinforcement learning on layer similarity, we perform reward-preference alignment using a reference model-free monolithic odds ratio preference optimization algorithm (ORPO), we utilize the fine-tuning framework enabled by Quantization and Low-Rank Adapters (QLoRA) Dettmers et al. (2023). A highly-curated and filtered set of DPO training datasets were compiled and made publicly available to produce the orpo-dpo-mix40k training set that was used for ORPO Hong et al. (2024) fine-tuning. Long examples were truncated to a maximum prompt length of 512 tokens for training. The models were limited to a max sequence length of 1024 tokens. Importantly, given the purpose of performing finetuning to explore any changes in the deeper layers, we limit fine tuning to 100 steps over one epoch using one thousand training samples, likely resulting in substantial early stopping. A linear learning schedule with a warm-up of 10 steps was used in combination with a linear rate of 8e-6. A key advantage of ORPO over traditional SFT and DPO is the low learning lates required. Loss optimization was performed with the huggingface paged adamw 8bit optimizer. On top of the Hugging Face Trainer API, we used quantization and Low-Rank Adapters (LoRA) Hu et al. (2022) for fine tuning in combination with ORPO. Quantization was performed with bitsandbytes library for QLoRA to 4bits. LoRA was performed using wih the Hugging Face peft library and dropout was set to

| Dataset Used | c4 | | UltraChat200k | | Chinese Medical Dialogue | |
|---|---|---|---|---|---|---|
| Block Size | Layers to Remove | Score | Layers to Remove | Score | Layers to Remove | Score |
| 1 | 25-25 | 0.121 | 25-25 | 0.11 | 24-24 | 0.109 |
| 2 | 24-25 | 0.151 | 24-25 | 0.137 | 23-24 | 0.143 |
| 3 | 23-25 | 0.178 | 24-26 | 0.16 | 24-26 | 0.168 |
| 4 | 24-27 | 0.203 | 23-27 | 0.182 | 24-27 | 0.188 |
| 5 | 23-27 | 0.225 | 23-27 | 0.204 | 23-27 | 0.211 |
| 6 | 22-27 | 0.248 | 23-28 | 0.225 | 23-28 | 0.232 |
| 7 | 22-28 | 0.267 | 22-28 | 0.248 | 22-28 | 0.251 |
| 8 | 20-28 | 0.289 | 20-28 | 0.273 | 21-28 | 0.276 |

Table 1: Average angular distance scores over 100 samples of input from three distinct datasets

0.05 with LoRA alpha of 32. Finally, target modules for parameter efficient fine tuning were applied to `['up_proj', 'down_proj', 'gate_proj', 'k_proj', 'q_proj', 'v_proj', 'o_proj']`. The described implementation of ORPO finetuning was largely based on a published guide accompanying the original manuscript introducing the new methodology of finetuning with ORPOHong et al. (2024).

## 5.4 Results

### 5.4.1 Determining the dependence of identified redundant layers on input text

Previous efforts aimed at identifying redundant layers in LLMs have relied on a single dataset as input. Indeed, limiting to one reference dataset allows for systematic analyses to determine the most high fidelity and predictive metrics for computing similarity across layers. Importantly, here we seek to establish whether the source and type of data used as input to measure perturbations across the network do influence the choice of which layers to prune. Accordingly, using 100 samples from 3 distinct datasets, we analyze differences in patterns of layer similarity. As previously shown, we also find that the deeper layers of the network tend to be more redundant with lower angular distance values. However, we do show for the first time that the choice of dataset used to inform pruning is important. We run a sweep of block sizes from 1 layer to 8 layers on the Llama3-8B model and find that the optimal layers are not globally recommended across all datasets (Figure 2, Table 1). More simply, we find that the most effective block to prune can vary. In a practical setting, where models are pruned to decrease requirements for fine-tuning, these findings suggest that the input dataset used to identify candidate layers should most closely align with the downstream supervised task. However, while the results suggest this is the ideal strategy, based on the average angular distance scores across the sweep, it is likely that using a single pruned model informed by a standard reference dataset (like c4) will not cause catastrophic impacts on other tasks.

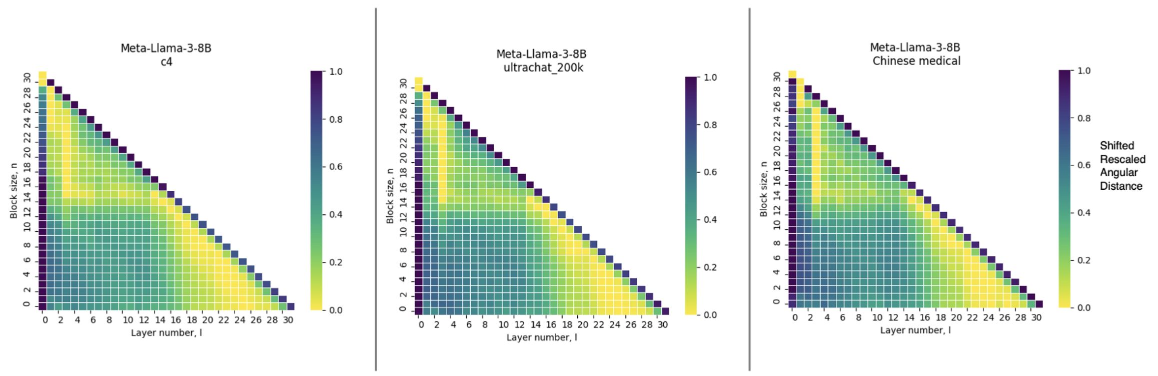


Figure 2: Normalized average angular distance between layers computed using 100 samples from three datasets representing distinct forms of text: c4 (left), UltraChat200k (middle), and Chinese Medical Dialogue (right).

| Model | Llama3-8B | Llama3-8B-minus3 | Llama3-8B-minus6 | Llama2-7B |
|---|---|---|---|---|
| HumanEvalPass@1 | 0.34 | 0.18 | 0.01 | 0.07 |

Table 2: Performance of coding capabilities of various versions of Llama model, included pruned models, as benchmarked on HumanEval dataset

### 5.4.2 Pruning of Llama3-8B

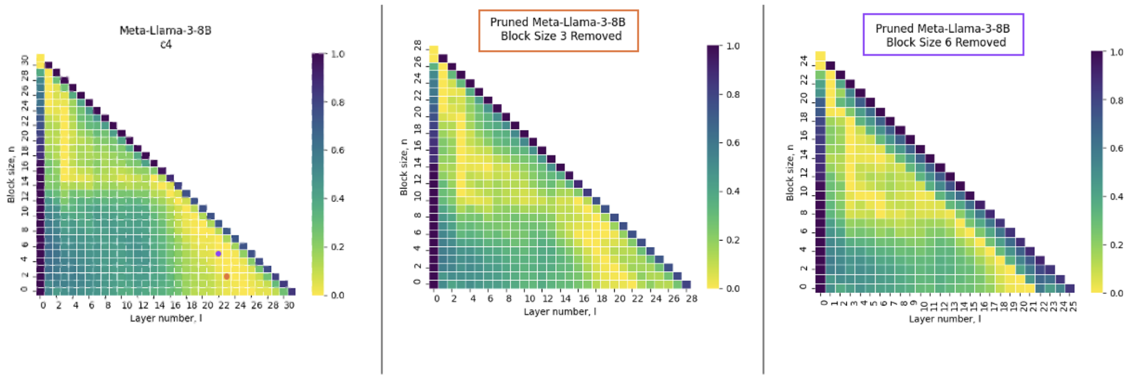


Figure 3: Angular distance between layers compared across models with 0, 3, and 6 pruned (from left to right) out of 32 total layers. Dots (orange and purple) are shown on the block selected for pruning to produce the 3-layer (orange) and 6-layer pruned models.

Using the angular distance score computed with the c4 dataset to identify the ideal layers for pruning, we pruned the Llama3-8B model to 90% and 80% of its original layer size (excluding the first and last layers) by pruning 3 and 6 layers, respectively. The corresponding models are referred to here as Llama3-8B-minus3 and Llama3-8B-minus6. We then computed the angular distance across the pruned model using c4 again to assess changes in angular distance patterns. We confirm that the resulting Llama3-8B-minus3 and Llama3-8B-minus6 models have substantially less blocks with nominal contribution (Figure 3).

### 5.4.3 Benchmarking Performance of a Pruned Model Across Coding Tasks

As previously described, using the Llama3-8B model, we identify the ideal starting layer for pruning of a 3 and 6 layer block, which correspond to dropping 10% and 20% of the total internal layers for the model. Using the pruned model, we then evaluated the coding function on the HumanEval benchmark as described above. The motivating study for this work showed that up to 40% of layers can be removed without affecting performance on broad general knowledge benchmarks like MMLU. Here, we find that even removing 10% of layers results in a nearly 50% drop in performance on coding. Dropping 20% of layers results in a nearly complete loss of coding capabilities. Interestingly, we find that compared with Llama2-7B, Llama3-8B-minus6 has over twice the Pass@1 score despite being approximately the same size (Table 2).

### 5.4.4 Fine-tuning after pruning

We next employed a form supervised fine tuning named ORPO: Monolithic Preference Optimization without Reference Model. A key advance of this method is the elimination of an additional preference alignment phase by introducing a minor penalty for disfavored generation that indirectly allows for preference alignment during supervised fine-tuning. While we recognize that a more rigorous analysis would compare the performance of the base model, fine tuned model, and fine tuned pruned model on public benchmarks (such as open-eval or light eval) to most definitively indicate the optimal recipe that balances performance with compute requirements, due to our own compute limitations, we instead consider the training loss.

As a proof of concept we fine-tune both Llama3-8B and Llama3-8B-minus6 using ORPO as described above on only a sample of 1000 model preference examples. However, in both cases, we see appreciable improvements in training with just 100 steps. Interestingly, we observe that the loss of the pruned model, which begins with poorer results, begins to approach the original loss of the unpruned model 4. While very preliminary, these findings are encouraging that performance may be able to be recovered with fine-tuning even after dropping 20% of the layers in the model.

## 6 Analysis

Pruning offers a promising method for rapidly reducing the memory footprint and requirements for performing inference and fine-tuning of large language models, and thereby democratizing access. In this study, we find that the identification of layers that are least impactful for pruning should be performed using a dataset that most closely aligns with downstream use. While qualitatively similar in the finding that the deeper layers are generally the least contributory, there are slight differences in the optimal set of layers that should be pruned based on the type of text used. This is an important notion given that models are increasingly be trained to be generalists during pre-training. For example, Llama3 supports over 30 languages.

However, the performance of emergent capabilities, which are the model's abilities that arise from scaling and are not directly modeled during pretraining, deteriorates rapidly with the removal of these layers. We find almost complete loss of coding after pruning only 20 % of layers. In contrast, others have shown that nearly 40 % of layers can be pruned with minimal impact on general knowledge tasks.

## 7 Conclusion

The collective results of this study suggest that while pruning can significantly enhance computational efficiency, it also risks compromising the model's performance, especially in tasks that require complex and nuanced understanding, such as coding. While this finding is limited to a single emergent capability and would benefit from a more comprehensive evaluation across tasks, this result alone is sufficient to urge caution when dropping layers – even those identified to play a less impactful role in prediction. Nonetheless, our benchmarking analysis also highlights the effectiveness of fundamental improvements during pretraining, such as tokenization strategies and dataset quality. We find that a pruned Llama3 model has over double the peformance on coding as an unpruned Llama2 of nearly equal size.

A key motivating question behind this work is whether a pruned large language model could be an attractive alternative to its original unpruned predecessor. This is particularly valuable in settings where specialized or resource-constrained hardware is not easily accessible. We also present preliminary data that supports the notion that finetuning on high quality datasets can help restore losses incurred during the pretraining process. In future work, it would be important to perform supervised fine tuning or model alignment until convergence to more rigorously support this claim.

## 8 Ethics Statement

While advances toward the development of more lightweight and easily deployable LLMs without the need for scarce and expensive specialized hardware will unequivocally be of considerable utility to society, it is critical to be aware of the potential concomitant risks. Misuse is a constant threat to progress toward increased accessibility. Thinner models with decreased compute requirements for finetuning can be more easily used for generating misinformation, hate speech, or other harmful content. Another major risk regarding small, yet powerful models, is related to increased privacy and security concerns. The ability to finetune and run inference on less secure systems, such as personal devices, could result in the inadvertent leak of sensitive information from private curated data.

Potential solutions that function to mitigate misuse may involve permanent robust content moderation and filtering systems to detect outputs that are potentially harmful. Other strategies may include traceability techniques that require checkpointing of each model iteration. Efforts to prevent leakage of private data could involve requiring all data to be encrypted and undergo anonymization prior to batching, or limiting finetuning to data stored in secure enclaves.

## References

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18.

Chris Brix, Parnia Bahar, and Volker Brix. 2022. Pruning transformers: Comparing structured techniques for weight pruning. *arXiv preprint arXiv:2202.07953*.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Joshua Luntungan Chong, Zeyi Wang, Qingxiu Shen, and Matthew Marge. 2022. Ultrachat-200k: Domain-specific open-domain conversational dataset for multitask evaluation. `https://huggingface.co/datasets/ultrachat/200k`.

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

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

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations (ICLR)*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. `https://huggingface.co/datasets/allenai/c4`.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2024. The unreasonable ineffectiveness of the deeper layers.

Stanford HAI. 2024. Hai index report 2024. `https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI_AI-Index-Report-2024.pdf`. Accessed: 2024-06-08.

Babak Hassibi and David G Stork. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171.

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

Jonathan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, David Hendricks-Arroyo, Juan Diego Rincon, Peter Lespiau, Jerry Williams, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shun Wang, Lu Chen, and Yanshuai Chen. 2022. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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

Yann LeCun, John S Denker, and Sara A Solla. 1990. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

Michael J. Mauboussin. 2023. Chatgpt and large language models: Six evolutionary steps. `https://blogs.cfainstitute.org/investor/2023/05/26/chatgpt-and-large-language-models-six-evolutionary-steps/`. Accessed: 2024-06-08.

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

Junjie Meng, Zhixu Wu, Siyuan Zhang, Lidong Zhang, Zhou Zhao, Xiaoling Liu, Minghui Bao, and Zhigang Ye. 2023. Chinese medical dataset. `https://huggingface.co/datasets/cmu-chinese-medical-dataset/chinese-medical-dataset`.

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

Anton Razzhigaev, Matvey Mikhalchuk, Elizaveta Goncharova, Nikolai Gerasimenko, Ivan Oseledets, Denis Dimitrov, and Andrey Kuznetsov. 2024. Your transformer is secretly linear.

Melisa Russak. 2024. Shorttransformers, optimal layer pruning tools.

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

Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, Fartash Faghri, Devang Naik, Oncel Tuzel, and Mohammad Rastegari. 2023. Weight subcloning: direct initialization of transformers using larger pretrained ones.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Goodman, Pavel Baudis, Jing Chiu, Chris Alberti, Maxime Gwinn, Yi Tay, Colin Raffel, et al. 2022. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Sam Shleifer. 2021. Low-resource language model distillation for conversational agents. *arXiv preprint arXiv:2104.08159*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297.

## A Appendix (optional)

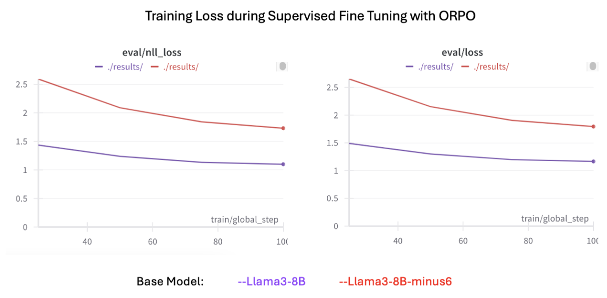
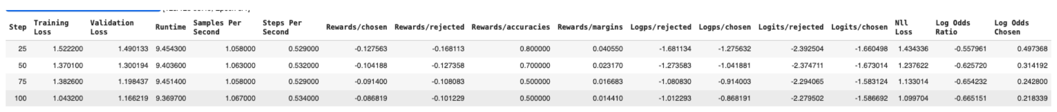
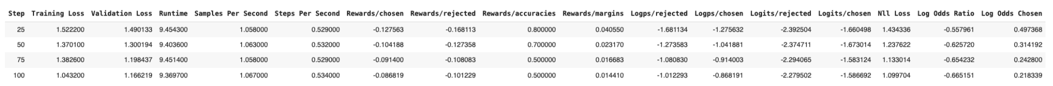


Figure 4: Log of training loss during supervised fine tuning with ORPO using either unmodified Llama3-8B (purple) or Llama3-8B with a block of 6 layers pruned (red)



| Step | Training Loss | Validation Loss | Runtime | Samples Per Second | Steps Per Second | Rewards/chosen | Rewards/rejected | Rewards/accuracies | Rewards/margins | Logps/rejected | Logps/chosen | Logits/rejected | Logits/chosen | Nll Loss | Log Odds Ratio | Log Odds Chosen |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 25 | 1.522200 | 1.490133 | 9.454300 | 1.058000 | 0.529000 | -0.127563 | -0.168113 | 0.800000 | 0.040550 | -1.681134 | -1.275632 | -2.392504 | -1.660498 | 1.434336 | -0.557961 | 0.497368 |
| 50 | 1.370100 | 1.300194 | 9.403600 | 1.063000 | 0.532000 | -0.104188 | -0.127358 | 0.700000 | 0.023170 | -1.273583 | -1.041881 | -2.374711 | -1.673014 | 1.237622 | -0.625720 | 0.314192 |
| 75 | 1.382600 | 1.198437 | 9.451400 | 1.058000 | 0.529000 | -0.091400 | -0.108083 | 0.500000 | 0.016683 | -1.080830 | -0.914003 | -2.294065 | -1.583124 | 1.133014 | -0.654232 | 0.242800 |
| 100 | 1.043200 | 1.166219 | 9.369700 | 1.067000 | 0.534000 | -0.086819 | -0.101229 | 0.500000 | 0.014410 | -1.012293 | -0.868191 | -2.279502 | -1.586692 | 1.099704 | -0.665151 | 0.218339 |



| Step | Training Loss | Validation Loss | Runtime | Samples Per Second | Steps Per Second | Rewards/chosen | Rewards/rejected | Rewards/accuracies | Rewards/margins | Logps/rejected | Logps/chosen | Logits/rejected | Logits/chosen | Nll Loss | Log Odds Ratio | Log Odds Chosen |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 25 | 1.522200 | 1.490133 | 9.454300 | 1.058000 | 0.529000 | -0.127563 | -0.168113 | 0.800000 | 0.040550 | -1.681134 | -1.275632 | -2.392504 | -1.660498 | 1.434336 | -0.557961 | 0.497368 |
| 50 | 1.370100 | 1.300194 | 9.403600 | 1.063000 | 0.532000 | -0.104188 | -0.127358 | 0.700000 | 0.023170 | -1.273583 | -1.041881 | -2.374711 | -1.673014 | 1.237622 | -0.625720 | 0.314192 |
| 75 | 1.382600 | 1.198437 | 9.451400 | 1.058000 | 0.529000 | -0.091400 | -0.108083 | 0.500000 | 0.016683 | -1.080830 | -0.914003 | -2.294065 | -1.583124 | 1.133014 | -0.654232 | 0.242800 |
| 100 | 1.043200 | 1.166219 | 9.369700 | 1.067000 | 0.534000 | -0.086819 | -0.101229 | 0.500000 | 0.014410 | -1.012293 | -0.868191 | -2.279502 | -1.586692 | 1.099704 | -0.665151 | 0.218339 |

Figure 5: Training metrics from 100 steps over one epoch using 1000 samples from orpo-dpo-mix40k dataset to perform ORPO fine-tuning of Llama3-8B base model (top) and Llama3-8B-Minus6 (bottom)