Gradual Attention Pruning for Effective LLM Inference

Anonymous ACL submission

Abstract

In this paper, we propose a simple yet effective pruning method for attention layers called Gradual Pruning (GP), which attempts to selectively prune attention heads in one stage at an adaptive pruning rate, without the need for later accuracy recovery. We investigate pruning rates from 25% to 75% at model scale from 7B to 13B parameters. Focusing on the attention layers offers substantial improvements in inference latency and integrates seamlessly with techniques such as quantization and token eviction. Practically, it requires minimal or no changes to existing inference frameworks, facilitating deployment. We evaluate our method on attention-intensive tasks like reasoning and in-context learning and provide insights for successful pruning.

1 Introduction

The advent of transformer-based Large Language Models (LLMs) has revolutionized the field of natural language processing, enabling remarkable language understanding capabilities (Brown et al., 2020; OpenAI et al., 2024; Gemini et al., 2024; Anthropic, 2024). These models leverage the self-attention mechanism (Vaswani et al., 2017) which enables easy access to previous generation states, yet at the price of explicitly keeping all previous attention key and value states cached in the memory. Repeated access to this ever-growing Key-Value (KV) cache during every step of generation poses a critical performance bottleneck.

Reduction of KV cache size is an active field of research. Many of methods, however, come with hidden deployment costs, such as extra experimentation or re-engineering existing inference pipelines. In contrast, simpler methods like pruning or sparsification offer a practical and often overlooked solution. Pruning removes less important components of the model, thereby reducing its size and computational requirements without significantly compromising performance (Michel et al., 2019). Unlike quantized models, pruned models can be safely and less expensively fine-tuned.

A leading pruning paradigm for transformer models is to jointly prune different components such as layers, attention heads, hidden dimensions, etc. (Xia et al., 2024; Muralidharan et al., 2024). However, attention pruning poses unique challenges; attention heads are activated dynamically, and involve learned position- and content-dependent interactions between heads (Voita et al., 2019; Jiang et al., 2024a). Consequently, identifying and removing less critical components without degrading model performance is harder than pruning neurons in feed forward layers, where the impact of pruning can be more easily isolated and estimated (Han et al., 2015). Another common aspect of pruning is operation in stages: (1) identifying and abruptly pruning redundancies, and (2) attempting to recover accuracy. In contrast, gradual pruning has proven effective (Zhu and Gupta, 2018), as it allows the smooth adaptation. Training LLMs, however, leaves little room for long adaptation schedules due to high costs.

In this paper, we investigate attention pruning in isolation from other model components and propose a novel method called Gradual Pruning (GP). Our approach applies a series of probabilistic gates using the Gumbel-Sigmoid distribution in an end-to-end fashion in order to non-uniformly prune entire attention KV heads. GP is guided by a simple algorithm that dynamically adjusts the pruning rate. GP offers competitive results to more common two-stage pruning, but also simplifies pruning hyper parameters, and produces a series of checkpoints pruned at different rates within a single run, reducing the need for iterative experimentation.

Lastly, we share insights on effective pruning of LLMs, gathered through a series of experiments. We make our code available at <*anonymized_url*>.

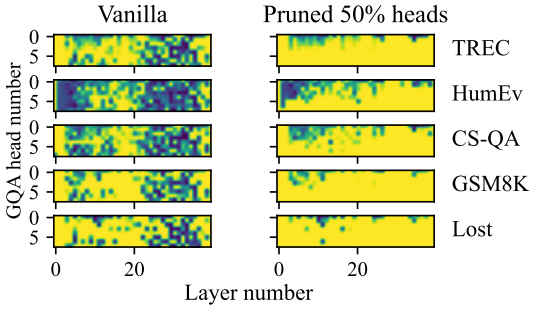

Figure 1: **Average attendance to the sink token** (navy) or anywhere else (yellow) in Mistral NeMo Minitron 8B. The data is sorted column-wise on TREC to show per-layer redundancy. The remaining tasks are sorted in the same order. In the base model, certain heads remain inactive regardless of the task. The pruned model (pruned heads colored yellow) rarely attends to sinks.

Algorithm 1: Adaptive adjustment of pruning loss scale α

Input: Target language modeling loss $\mathcal{L}_{target}$,
Attention pruning target H_{target},
Constants $d > 1$ and $s > 0$
Initialize: $\alpha \leftarrow 1$, $\mathcal{L}_{EMA} \leftarrow \mathcal{L}_{target}$
while *true* **do**
  $\mathcal{L} \leftarrow$ training_step()
  $\mathcal{L}_{EMA} \leftarrow \gamma \cdot \mathcal{L}_{EMA} + (1-\gamma) \cdot \mathcal{L}$
  if $\mathcal{L}_{EMA} < \mathcal{L}_{target}$ **then**
    $H_{target} \leftarrow 0$ `// Reset the target`
    $\alpha \leftarrow \alpha + s$
  else
    if $H_{target} = 0$ **then**
      $\alpha \leftarrow \alpha / d$ `// Decrease once`
      $H_{target} \leftarrow H_{current}$ `// Freeze the target`

2 Background

Multi-head Self-attention. We focus on multi-head self-attention with a causal mask. It takes as input a matrix $X = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^{n,d}$ of hidden states x_i corresponding to input tokens and linearly projects each hidden state into h query, key, and value vectors $Q_i = XW_i^Q$, $K_i = XW_i^K$, $V_i = XW_i^V$. Then, for each pair of embeddings, an inner product is calculated

$$S_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d/h}} + M\right), \qquad (1)$$

where M is an upper triangular masking matrix. The final result is $Y_h = [S_1 V_1, \ldots, S_h V_h] W^O$.

Multi-stage pruning. We choose Sheared Llama (Xia et al., 2024) as an example of a method which follows multi (two) stage pruning process: first, the model is pruned, and then the model undergoes further training to regain accuracy. The model includes a series of probabilistic variables that act as gates for entire layers, hidden dimension, attention heads and intermediate dimensions of feed-forward modules. *Uniform* pruning, i.e., pruning the same number of heads and dimensions in every layer, is enforced through Lagrange multipliers. Additionally, a sampling algorithm for dataset subsets is applied in attempt to make the training more data-efficient.

Attention redundancies. Analysis of attention attendance patterns insights into the kind of computations performed by the attention mechanism (Olsson et al., 2022; Wu et al., 2024) and helps identify potential redundancies (Jiang et al., 2024b). We focus on the content-dependent nature of these patterns, and examine attention to the sink token (Xiao et al., 2024), noting that attending to the beginning-of-sentence token makes the computation entirely context-free, therefore reducible.

The patterns are asymmetrical (Figure 1), with some tasks being more attention-intensive than others, providing motivation for non-uniform pruning. Interestingly, some heads remain fairly inactive regardless of the task. In contrast, a pruned model, detailed in Section 4.1, shows very little attention sink activity for the most attention-intensive tasks.

3 Gradual Pruning

We build on the *retrofitting* approach (Nawrot et al., 2024), where Gumbel-Sigmoid gates are added to the model with logits initialized to small positive values to begin training with open gates. The temperature τ is set to a small value and kept constant throughout the entire training, which helps allocate most of the probability mass to either 0 or 1.

Let h be the number of value heads in a single attention layer. In every attention layer, we add a trainable vector $\boldsymbol{g_h} \in \mathbb{R}^h$, which is later used as log-probabilities in Gumbel-Sigmoid and interpreted as gates. Below, we drop the layer index for clarity:

$$\begin{aligned} \boldsymbol{g'_h} &= \text{GSIGMOID}\left(\boldsymbol{g_h}, \tau\right), \\ \boldsymbol{V} &= (\boldsymbol{X}\boldsymbol{W}_v) \circ \boldsymbol{g'_h}. \end{aligned} \qquad (2)$$

In order to induce pruning, we add an additional loss term $\mathcal{L}_p$ to the language modeling loss:

$$H^l_{used} = \sum_{i=1}^{h} \left[(\boldsymbol{g'_h})_i > 0.5\right],$$

$$\mathcal{L}_p = \alpha \cdot \max\left(0, -H_{target} + \sum_{l \in layers} H^l_{used}\right),$$

	Method	KV Cache Size	MMLU 5-shot	HumEv pass@1	CS-QA 0-shot	GSM8K 8-shot
Llama 2 7B	Fine-tuned	100%	48.0	18.3	70.3	14.0
	Gradual	50%	**46.0**	12.8	69.2	9.1
	Two-stage	50%	45.1	**13.4**	**69.5**	**11.3**
	Two-stage uniform	50%	36.8	10.4	68.6	8.9
	Importance	50%	33.3	9.8	68.2	6.4
	GQA	50%	34.8	12.8	67.7	8.6
	Gradual	25%	**32.5**	10.4	**66.6**	5.6
	Two-stage	25%	29.6	**11.0**	**66.6**	**5.9**
	Two-stage uniform	25%	28.9	9.8	64.6	4.2
	Importance	25%	27.9	9.8	66.1	3.5
	GQA	25%	26.9	9.8	65.9	4.8
Llama 2 13B	Fine-tuned	100%	56.8	17.7	73.2	27.5
	Gradual	50%	**55.6**	15.8	**73.0**	**23.6**
	Two-stage	50%	55.0	16.5	72.9	23.0
	Two-stage uniform	50%	52.0	**17.7**	71.8	17.3
	Importance	50%	52.9	16.5	71.6	18.0
	GQA	50%	49.9	15.2	71.5	17.2
	Gradual	25%	**51.2**	14.6	**71.8**	**12.0**
	Two-stage	25%	50.2	**15.2**	70.3	10.5
	Two-stage uniform	25%	43.7	9.8	66.0	6.8
	Importance	25%	45.1	11.6	68.6	7.8
	GQA	25%	41.4	**15.2**	69.1	9.9

Table 1: Downstream task accuracy of pruned models. `Gradual` and `Two-stage` prune entire heads, and `Two-stage uniform` attempts to prunes the same number of heads in every layer. Gradual Pruning obtains the highest MMLU scores across all tested methods.

where α is a scaling factor and $[x > 0.5]$ is the Iverson notation, that is $[x > 0.5] = 0$ when $x \leq 0.5$ and 1 otherwise. As this operation is not differentiable, we apply a straight-through gradient estimator, so in our case $\frac{\partial[f(x)>0.5]}{\partial x} = \frac{\partial f(x)}{\partial x}$.

The goal of $\mathcal{L}_p$ is to ensure the pruning target H_{target} is achieved. Notably, the gradient of this loss is independent of the model's distance from the target. Our pruning criterion is *non-uniform*, allowing adaptive pruning of different numbers of heads in each layer. As discussed in Section 2, this approach is expected to provide better quality. We discuss its performance in Section 5.

3.1 The Scheduling Algorithm

In the approach by Nawrot et al. (2024), the scale of the added compression loss term remains constant during training, with gradual compression achieved by increasing the target compression value. However, we faced challenges in adjusting the loss scale α: small values delayed achieving the pruning target, while large ones degraded model quality. Additionally, using logit distillation (Hinton et al., 2015) required new adjustments of α. To manage both α and H_{target} without excessively prolonging training, we incorporated a solution inspired by the Adaptive Gain Control (AGC) algorithm.

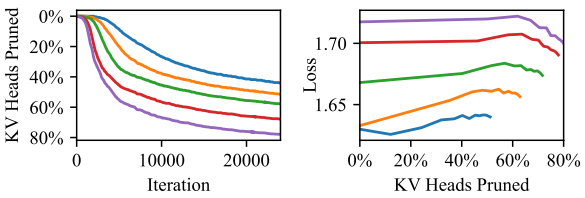

Figure 2: Llama 2 7B pruning runs with different loss threshold $\mathcal{L}_{target}$. The threshold is straightforward to set based on the evaluation perplexity on the target dataset.

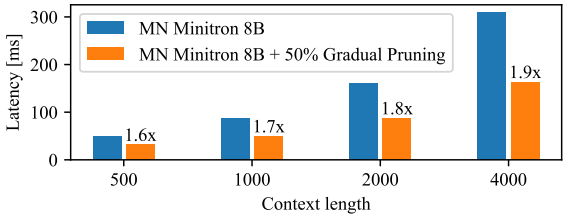

Figure 3: Inference latency of token generation with batch size 64 in 16-bit precision measured on NVIDIA H100 GPU using Megatron-LM framework.

AGC dynamically adjusts parameters in response to changes in the input in order to maintain a desired output. Noisy input can be smoothed, e.g., with exponential moving average (EMA). In our context, we calculate the EMA of the language modeling loss $\mathcal{L}_{EMA}$ and adapt the pruning rate accordingly (Algorithm 1). If $\mathcal{L}_{EMA}$ exceeds the predefined threshold, H_{target} is frozen. The model continues to explore different pruning patterns until $\mathcal{L}_{EMA}$ falls below the threshold. Figure 2 shows how different loss thresholds influence pruning rate and the loss during pruning.

4 Experiments

Model families. We apply pruning to Llama 2 7B and 13B (Touvron et al., 2023b), GQA-enabled Llama 3.1 8B (Dubey et al., 2024), and Mistral NeMo Minitron 8B (Muralidharan et al., 2024).

Training setup. We adapt training parameters from (Touvron et al., 2023a) and lower the batch size from 1024 to 128 with a lower learning rate for better data efficiency (Tworkowski et al., 2023) for all tested methods. For Gumbel-Sigmoid, we use a constant temperature $\tau = 0.1$. In order to mitigate dataset and training setup mismatch and best preserve the reasoning abilities, we train the 8B models with logit distillation similar to (Muralidharan et al., 2024) (for details see Appendix C).

Model	Method	KV Cache Size	MMLU 5-shot	TREC 180-shot	HumanEv pass@1	CS-QA 0-shot	GSM8K 8-shot	Lost
Llama 3.1 8B	Vanilla	100%	67.1	89.6	36.6	75.0	53.1	100.0
	Gradual	75%	66.1	85.0	32.9	74.1	49.4	98.5
	Gradual	50%	65.0	86.6	36.0	74.1	49.0	100.0
Mistral NeMo Minitron 8B	Vanilla	100%	69.5	89.4	37.2	76.8	59.7	100.0
	Gradual	80%	68.7	87.6	35.4	76.5	58.4	100.0
	Gradual	70%	68.5	91.6	32.3	75.9	57.5	99.6
	Gradual	60%	67.7	85.6	31.1	75.7	56.3	99.8
	Gradual	50%	67.5	84.4	29.9	75.6	57.0	99.8

Table 2: Downstream task accuracy for GQA-enabled models pruned gradually with logit distillation. Even after pruning half of attention heads both models retains high scores. All models achieve nearly perfect performance in Lost in the Middle benchmark, but show decaying in-context learning performance in the TREC task.

Methods. We experiment with Sheared Llama variants, denoted `Two-stage`, applying probabilistic gates only to attention heads, with and without the uniformity constraint. In preliminary experiments, gating entire attention layers led to sub-optimal pruning, as the algorithm quickly chose entire layers, resulting in a local minimum hard to escape (see Appendix E). Thus, we avoided gating entire layers in subsequent experiments.

We include two additional baselines: Grouped Query Attention (GQA; Ainslie et al., 2023) and greedy importance-based pruning Muralidharan et al. (2024), which produce uniform architectures.

Datasets. We train on curated dataset composed of open source data (`<citation_anonymized>`).

4.1 Results

We describe downstream tasks in Appendix B.

Pruning models without GQA. Table 1 shows the results of pruning Llama 2 models. The GQA baseline and importance pruning perform significantly worse than `Gradual` and `Two-stage` methods for the 13B model, with an even more pronounced difference for Llama 2 7B. Predictably, uniform models, which prune the same number of heads in every layer, perform significantly worse than their non-uniform counterparts. All tested methods show noticeable regressions on the GSM8K benchmark and HumanEval. `Gradual` consistently scores best on MMLU. We hypothesize that gradual training better preserves the weights and factual knowledge stored in feed-forward layers. Pruning 50% of attention heads removes 1.1B and 2.1B parameters from the 7B and 13B models, respectively.

Pruning aided by distillation. Finally, we present experiments on pruning GQA-enabled Llama 3.1 8B and Mistral NeMo Minitron 8B with logit distillation (Table 2). Despite the severity of pruning and the models' sensitivity to different training data, they exhibit strong performance on downstream tasks. The largest accuracy drops occur in the HumanEval benchmark, partly due to artifacts like extra whitespace in the prompts. Mistral NeMo Minitron 8B, already pruned from a 12B model, shows that pruning 30%, 40%, and 50% of KV attention heads results in reductions of 0.5B, 0.67B, and 0.84B parameters, respectively. Lastly, the inference latency of the pruned models improves by up to 2× as context length increases (Figure 3). Throughput gains are even higher, as freeing KV cache memory allows for larger batch sizes.

5 Related Work

Pruning is a well-established technique for improving Transformer LLM inference efficiency (Wang et al., 2024; Zhou et al., 2024). LLM pruning can be categorized by (1) structured vs. unstructured pruning, and (2) whether post-pruning fine-tuning is required. Unstructured pruning, such as removing individual weights (Frantar and Alistarh, 2023), offers flexibility but is uncommon due to technical difficulties in using sparse structures. Structured pruning of entire layers or attention heads often leads to quality degradation (Zhong et al., 2024).

Several works have applied structured pruning to LLMs, followed by training to recover the lost performance (Ashkboos et al., 2024; Ma et al., 2023; Lv et al., 2024). However, these methods prune based on predetermined criteria, which might restrict adaptability and lead to slower convergence.

Similar approaches (Huang et al., 2024; Liu et al., 2024) have explored gradual pruning but either prune entire layers or apply uniform pruning across layers, which is suboptimal for attention head pruning (Xia et al., 2022). GP specifically targets attention heads and allows for heterogeneous patterns, enabling more effective pruning.

Limitations

Training large language models (LLMs) is complex, with many critical details often omitted. Here, we outline the main limitations of Gradual Pruning (GP).

Model sizes and families. We have tested our method on Llama 2, Llama 3.1 and Mistral NeMo Minitron models ranging from 7B to 13B parameters. However, models that are one or two orders of magnitude larger may behave differently. Future work will explore larger models and different families.

Tensor parallelism. Tensor parallelism, which splits operations across multiple accelerators, is common in LLM inference. For multi-head attention layers, this split typically occurs over the head dimension. Non-uniformly pruned models must have a head count divisible by the number of accelerators to maintain performance benefits. If tensor parallelism hinders performance, GP can be run twice: first to identify pruning opportunities, and second with constraints to guide pruning.

Long context accuracy. Pruning attention heads might impact long-context retrieval. We have not conducted rigorous evaluations to ensure long-context capabilities are preserved, except for the Lost in the Middle downstream task.

Influence of the dataset. Our final models were trained with logit distillation to lessen the influence of shifting the training dataset. However, we have not evaluated the impact of different datasets on the effectiveness of pruning.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *Preprint*, arXiv:2305.13245.

Anthropic. 2024. Claude 3.5.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *ArXiv*, abs/2401.15024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, et al. 2020. Language models are few-shot learners. *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, et al. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *ArXiv*, abs/2301.00774.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, et al. 2024. A framework for few-shot language model evaluation.

Team Gemini Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, et al. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1135–1143, Cambridge, MA, USA. MIT Press.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. 00031.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Weiyu Huang, Guohao Jian, Yuezhou Hu, Jun Zhu, and Jianfei Chen. 2024. Pruning large language models with semi-structural adaptive sparse training. *ArXiv*, abs/2407.20584.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024a. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024b. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Preprint*, arXiv:2407.02490.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023b. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Yijiang Liu, Huanrui Yang, Youxin Chen, Rongyu Zhang, Miao Wang, Yuan Du, and Li Du. 2024. Pat: Pruning-aware tuning for large language models. *ArXiv*, abs/2408.14721.

Bo Lv, Quan Zhou, Xuanang Ding, Yan Wang, and Zeming Ma. 2024. Kvpruner: Structural pruning for faster and memory-efficient large language models. *ArXiv*, abs/2409.11057.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *ArXiv*, abs/2305.11627.

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

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *Preprint*, arXiv:2407.14679.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. 2024. Dynamic memory compression: Retrofitting llms for accelerated inference. *Preprint*, arXiv:2403.09636.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. 2022. In-context learning and induction heads. *CoRR*, abs/2209.11895.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2023. Focused transformer: Contrastive training for context scaling. *Preprint*, arXiv:2307.03170.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

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

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *ArXiv*, abs/2402.09748.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval head mechanistically explains long-context factuality. *ArXiv*, abs/2404.15574.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared llama: Accelerating language model pre-training via structured pruning. *Preprint*, arXiv:2310.06694.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. *ArXiv*, abs/2204.00408.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

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

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. 2024. Blockpruner: Fine-grained pruning for large language models. *ArXiv*, abs/2406.10594.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. 2024. A survey on efficient inference for large language models. *ArXiv*, abs/2404.14294.

Michael H. Zhu and Suyog Gupta. 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression.

A GP Pseudocode

Listing 1: Pseudocode of GP training

```
def gp_apply(values, gate_head, temperature):
    batch, seq, head, head_dim = values.shape
    assert gate_head.shape == (head,)

    # by initializing gates logits to 5/lr_scale and then multiplying them
    # by lr_scale we can increase the speed of gate learning without increasing
    # the learning rate of the model. We find that in general it suffices to
    # scale the learning speed of the gates so that they can open and close during
    # the tuning. In Sheared Llama learning rate of gates is altered at the level
    # of the optimizer
    gate_head = gate_head.broadcast_to(batch, head) * gate_lr_scale

    gate_head_soft, gate_head_discrete = gumbel_sigmoid(gate_head / temperature)

    gate_value_soft = gate_head_soft.reshape(batch, 1, head, 1)
    gate_value_discrete = gate_head_discrete.reshape(batch, head)

    values = values * gate_value_soft

    head_usage_percent = gate_value_discrete.mean()

    return values, head_usage_percent


# for simplicity we assume that each layer has the same number of heads
def calc_pruning_loss(per_layer_head_usage_percent, target_heads_percent):
    # mean across the layers
    head_usage_percent = per_layer_head_usage_percent.mean()
    return max(0, head_usage_percent - target_heads_percent)


def calc_loss(lm_loss, per_layer_head_usage_percent):
    target_head_usage_percent, prunning_loss_scale = adals(lm_loss)
    pruning_loss = calc_pruning_loss(
        per_layer_head_usage_percent, target_head_usage_percent
    )

    return lm_loss + prunning_loss_scale * pruning_loss


for inputs, targets in data:
    logits, per_layer_head_usage = model(inputs)
    lm_loss = calc_lm_loss(logits, targets)
    loss = calc_loss(lm_loss, per_layer_head_usage)
    optimizer.update(model, loss)
```

B Downstream Task Details

We test our models on a wide selection of downstream tasks, that measure knowledge, reasoning, coding, long context manipulation, and mathematical abilities of the models. Here we briefly describe the tasks.

MMLU (Hendrycks et al., 2021) - is a collection of tests spanning multiple subjects across humanities, STEM, and social sciences. The full list of domains is as follows: abstract algebra, anatomy, astronomy, business ethics, clinical knowledge, college biology, college chemistry, college computer science, college mathematics, college medicine, college physics, computer security, conceptual physics, econometrics, electrical engineering, elementary mathematics, formal logic, global facts, high school biology, high school chemistry, high school computer science, high school european history, high school geography, high school government and politics, high school macroeconomics, high school mathematics, high school microeconomics, high school physics, high school psychology, high school statistics, high school us history, high school world history, human aging, human sexuality, international law, jurisprudence, logical fallacies, machine learning, management, marketing, medical genetics, miscellaneous, moral disputes, moral scenarios, nutrition, philosophy, prehistory, professional accounting, professional law, professional medicine, professional psychology, public relations, security studies, sociology, us foreign policy, virology, world religions. We utilize LM Eval Harness framework (Gao et al., 2024) in 5-shot setup. As the score of the model, we calculate the average across all subjects.

TREC (Li and Roth, 2002; Hovy et al., 2001) - is a question classification dataset, where each question needs to be classified in one of six classes: description and abstract concepts (DESC), entities (ENT), abbreviation (ABBR), human beings (HUM), numeric values (NUM), and locations (LOC). We utilize LM Eval Harness framework (Gao et al., 2024), 180 shot and evaluate in multiple-choice setup. Note that in this task models need to learn labels (DESC, ENT, ABBR) in context. Examples

Q: When was Ozzy Osbourne born ?
A: NUM
Q: Name the scar-faced bounty hunter of The Old West.
A: HUM
Q: What sprawling U.S. state boasts the most airports?
A: LOC

HumanEval (Chen et al., 2021) - in this task, the model is given a prefix of a function along with a comment, and it needs to complete the function. We utilize LM Eval Harness framework (Gao et al., 2024) in 0 shot setup with 512 token limit for code generation and framework from (Liu et al., 2023a) to execute the code and provide pass@1 results.

ARC-Challenge & ARC-Easy (Clark et al., 2018) - contain multiple-choice questions from science exams from grades 3 to 9. For evaluation, we utilize the LM Eval Harness framework (Gao et al., 2024) in a 0 shot setup.

BoolQ (Clark et al., 2019) - is a collection of yes/no questions from the Google search engine. For evaluation, we utilize the LM Eval Harness framework (Gao et al., 2024) in a 0 shot setup.

HellaSwag (Zellers et al., 2019) - in this task model needs to choose the most probable continuation of a given passage. For evaluation, we utilize the LM Eval Harness framework (Gao et al., 2024) in a 0 shot setup.

PIQA (Bisk et al., 2019)- test the physical commonsense reasoning of the models, by asking them to choose the most fitting solution given the problem. For evaluation, we utilize the LM Eval Harness framework (Gao et al., 2024) in a 0 shot setup.

WinoGrande (Bisk et al., 2019)- test commonsense reasoning in case of pronoun resloution. For evaluation, we utilize the LM Eval Harness framework (Gao et al., 2024) in a 0 shot setup.

Lost in the Middle (Liu et al., 2023b) - Test the ability of the model to retrieve information from various places in context. We utilize the key value version and evaluate our model generations using LM Eval Harness framework (Gao et al., 2024) in a 0 shot setup. We prompt the model as in the example below and use the following regex (([a-f0-9])8-([a-f0-9])4-([a-f0-9])4-([a-f0-9])4-([a-f0-9])12) to retrieve the

answer. We evaluate retrieval at positions {0, 5, 10, 15, 20, 25, 30, 35, 40, 44, 49} and average the scores to get the final accuracy. For evaluation, we utilize the JSON files provided by the benchmark authors.


```
{"635a8830-94a6-446d-8d45-0f74349debdb": "f457293f-e2b6-4fde-915b-56967e1eff75",
"48b11c76-eec7-4b7d-9c62-e869b230cfa4": "2deb4a2f-3a48-4050-95f8-4ff7c01de229",
"36e82bc1-48a4-427b-a894-ae1ef0816d89": "c722c0f4-7755-4d66-b26b-b1efd5ca617b",
"ea41fa62-0523-4498-922d-22d622c9b5db": "87063bd9-c4e7-47eb-ad5a-d83e6805d2a8",
"0ff9d44b-c695-4b36-9ca7-d009fc6902f4": "30cf7d49-2e14-470d-a270-a45a63be5697",
"6dcd0360-a58d-4981-8251-d1a2d8e9643c": "881065a9-6228-4702-a3a8-6bfc25a7da29",
"d5e6d55e-4d0d-4767-80ce-52fbaf342dad": "0f3f1e5b-e46e-4186-83b1-df57feb0ae40",
"0d43bd11-5a92-4acb-a370-b19863e954b5": "e8862760-a4c9-4cf3-b38e-f2cd1bd18c60",
"63a609a9-faced-4bce-b6d3-63d4547c3387": "9875c474-1c03-4c9c-9e21-f7620b568673",
"1a04f308-2448-494d-b6e0-1ce0c5dfcafe": "b9b846ba-553a-4fa8-8a70-a02da4e9c5c2",
"07595143-e052-4f10-babd-abe168707490": "23d29d1f-97ee-43a4-8509-769809fdd3d9",
"f79e0a50-dcb4-4160-a877-62fdf9dd818c": "2327b6ac-5fd5-4133-bc48-e1a5454161dc",
"cb22e8d5-f136-4b96-a929-6665f9738bb3": "c2703104-29fc-4f74-bd6f-a84509e16247",
"d7a2a172-6f3a-4c73-866b-22faf043c54e": "976b0210-dd10-40fa-8ec7-050dc92d8fbf",
"0179edd4-2c14-4f70-b8c2-7d76ea79c7f8": "22a4b952-0ab4-4487-a0a7-ffc4f6e32045",
"fe7a3b03-a81f-40de-b962-0cd0a894c884": "efdd5fd8-4124-4055-93d4-5a32d509f230",
"6f6de911-b298-4b46-9617-ebb66a3c287f": "c18f2ba0-c50c-402f-bd02-faa566dbddc9",
"29e714a4-619c-423b-b05f-ae6aef20034d": "c8580b62-771d-4101-8fd1-fc0db4fc8e4f",
"7aeefc1c-615f-4173-a80a-0b46f2333702": "157b5ad3-c4c0-437b-b782-df9aeca6b640",
"44f3ccaf-7da7-4c44-a615-8a319735d4e7": "3be520a3-6ccd-4f73-9860-19f70f12aa8c",
"c923440b-d231-4065-a922-08442d47c6aa": "67acf73f-ce3d-448f-a6d6-b4ba7917eb33",
"43ca4f58-a8f4-494f-9ccf-8f8e581bbc90": "0f4e64f2-0aa9-4127-b501-a0e9f95976a9",
"634d6ed2-5d67-4989-b694-25be98286dae": "5ca513b3-f404-451b-9a5c-8b2efce0a178"}
"29e714a4-619c-423b-b05f-ae6aef20034d":
```


GSM8K (Cobbe et al., 2021) - is a collection of grade school math problems. We utilize LM Eval Harness framework (Gao et al., 2024) in a 8 shot setup with examples from (Wei et al., 2023) and 256 token generation limit.

C Hyperparameter and Model Details

In Table 3 we show AGC parameters that we use tune our models. For Two-Stage we have tested $\{0.01, 0.1, 1.0\}$ learning rates of Sheared LLaMA gates and picked the one that achieved the desired pruning within 4K steps while keeping the lm-loss low (0.1). For GP we have chosen to scale the gate values gate_lr_scale $= 300$ (see Appendix A).

Tuning Parameters				
Model	LLaMA 2 $7B$	LLaMA 2 $13B$	LLaMA 3.1 $8B$	Minitron $8B$
Common				
Num Layers	32	40	32	40
Hidden Dim	4096	5120	4096	4096
FFN Dim	11008	13824	14336	11520
Vocab Size	32K	32K	128K	131K
Init Num Query Heads	32	40	32	32
Init Num KV Heads	32	40	8	8
Learning Rate	1e-5	1e-5	1e-5	1e-5
Batch Size	128	128	128	128
Context Length	4096	4096	8192	8192
Tuning Tokens	12.5B	12.5B	12.5B	12.5B
AGC & GP				
Temperature τ	0.1	0.1	0.1	0.1
LM Loss Threshold	1.66	1.59	1.96	1.68
LM Loss EMA γ	0.2	0.2	0.2	0.2
Pruning Loss Scale Decay d	0.75	0.75	0.75	0.75
Pruning Loss Scale Step s	0.06	0.06	1.5	1.5
Sheared LLaMA				
Start Sparsity	0.0	0.0	-	-
Target Sparsity	0.75	0.5	-	-
Lagrangian Warmup Steps	320	320	-	-
Lambdas LR Scale	-10K	-10K	-	-
Gates LR Scale	10K	10K	-	-

Table 3: Parameters used in model tuning; For Sheared LLaMA we adopt parameters from the original paper and grid lr scales.

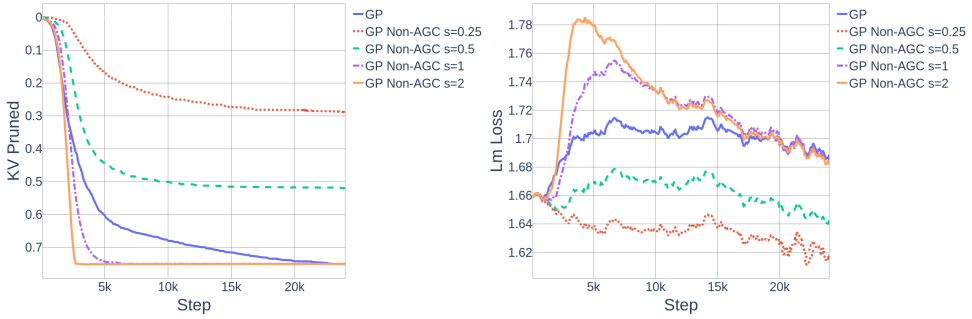

Figure 4: Comparsion of lm-loss and head removal during GP pruning with and without AGC. In each experiment we train a 7B parameter Llama 2 model for 24k steps with batch size 128 and learning rate $1e-5$ with target set to removing 75% of attention heads. For AGC we use 1.72 as a threshold (see Algorithm 1).

D Ablations

D.1 Benefits of AGC

In this section, we study the benefits of AGC. To be more precise, we tune 7B parameter Llama 2 models with GP, but without AGC, we uses a fixed pruning loss scale, and the target set to prune 75% of the whole attention heads. We compare them to models tuned with GP and AGC. We note that the approaches tuned without AGC tend to create a loss spike that has been observed to result in model degradation (Nawrot et al., 2024). In particular, (Nawrot et al., 2024) observed that a spike in LM loss can degrade a model's downstream performance such that it does not recover, even after returning to the original (or achieving a lower) LM loss. In Figure 4, we show how the LM loss changes during the tuning procedure. We note that AGC can reliably keep the lm loss below the predefined 1.72 threshold, whereas other methods result in lm loss explosion.

D.2 Hyperparameter robustness

In this section, we test the hyper-parameter robustness of AGC and GP. For AGC, we test different values of the lm-loss/downstream performance threshold t, scale step s, scale decay d, and EMA γ (see Algorithm 1). For GP, we test the importance of the gating temperature τ and adopt gates initialization from (Nawrot et al., 2024). We note that the most important parameter is the performance threshold t, which specifies the lower bound on model performance (upper bound on lm loss). We note that the AGC scale parameter (d) should be set to a value < 1, as otherwise, the algorithm cannot directly reduce pruning loss scale, which results in the quick removal of almost all attention heads and a huge loss spike. Other parameters have reasonable tolerance ranges and behave monotonically. We shot the results in Figures 5 and 6.

For all experiments, we use 7B parameter Llama 2 models and prune channels of key and value heads independently (instead of whole heads). That is instead of creating a gater per value head we create gate for each value and key head dimension (separate for key and values). We test each parameter separately, setting the remaining parameters as follows: $t = 1.66$, $s = 0.06$, $d = 0.75$, $\gamma = 0.5$, $\tau = 0.1$.

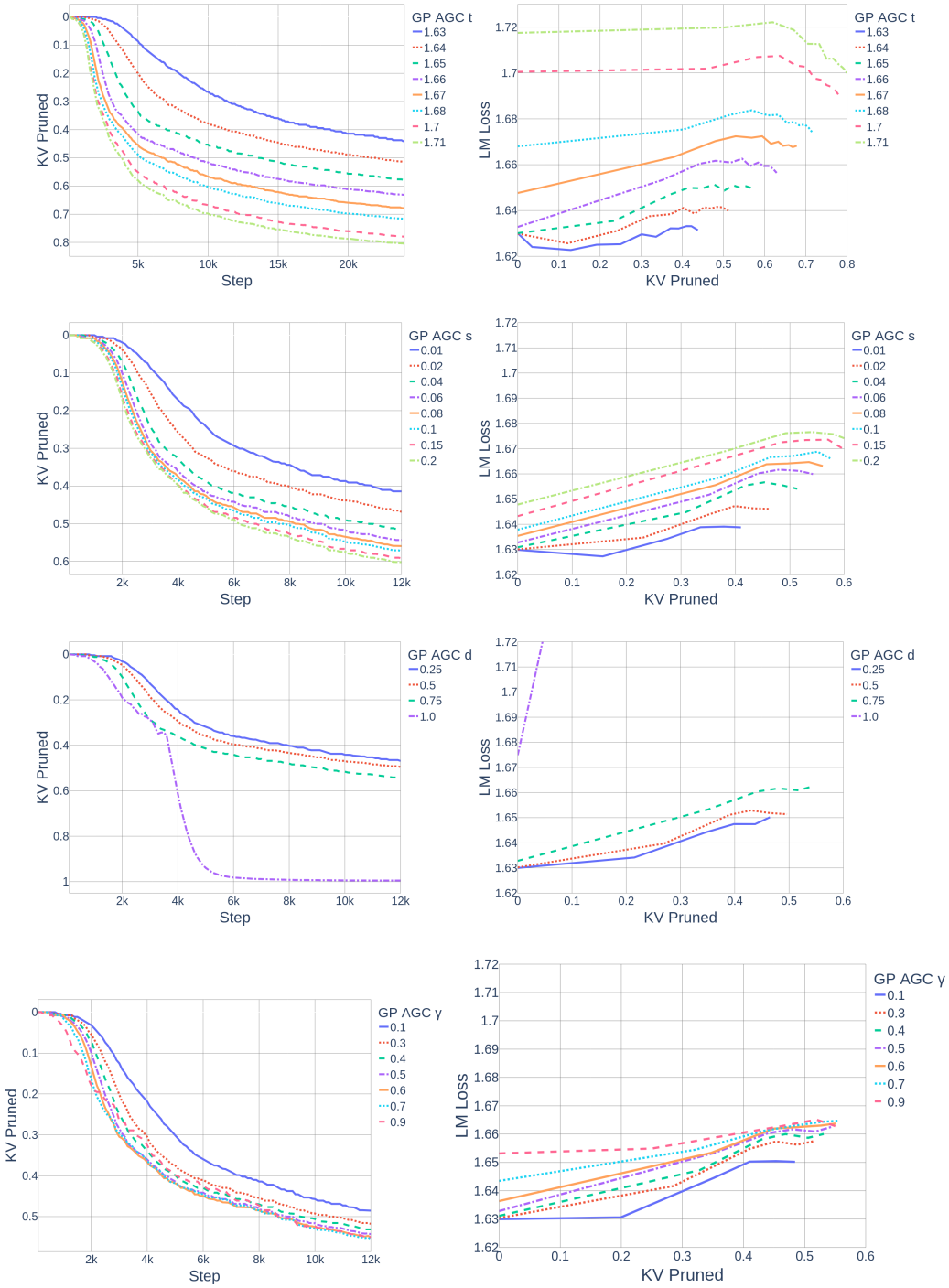

Figure 5: AGC parameters and their influence on compression speed. As one can observe the effects are monotonic. That is increasing one parameter results in increase/decrease in the pruning speed. In each experiment (except the loss threshold one) we train a 7B parameter Llama 2 model for 12k steps with batch size 128 and learning rate $1e-5$.

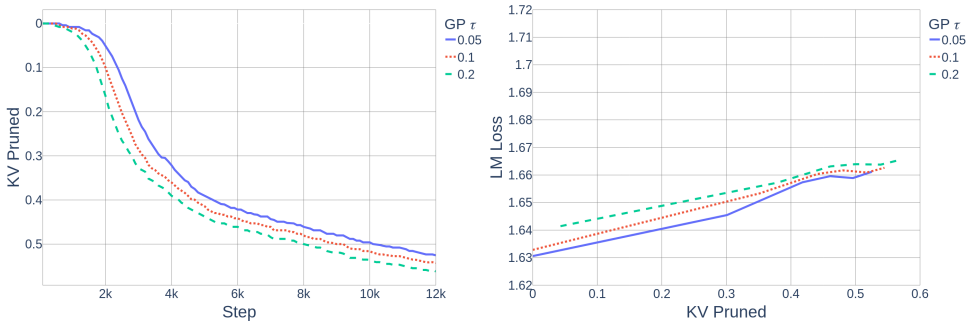

Figure 6: Influence of τ parameter on GP performance. Smaller temperature τ sharpens the output of GUMBELSIGMOID. In each experiment we train a 7B parameter Llama 2 model for 12k steps with batch size 128 and learning rate $1e-5$.

E Pruning Heads and Attention Layers

Model	Method	KV Cache Size	MMLU 5-shot	HumanEv pass@1	CS-QA 0-shot	GSM8K 8-shot	Lost 0-shot
Llama 2 7B	Two-stage	50%	45.1	13.4	69.5	11.3	83.8
	Two-stage + attn layers	50%	40.5	13.4	68.0	8.4	81.2
Llama 2 13B	Two-stage	50%	55.0	16.5	72.9	23.0	88.0
	Two-stage + attn layers	50%	52.5	16.5	71.9	15.2	93.6

Table 4: Downstream task accuracy of pruned models. `Gradual` and `Two-stage` prune entire heads, and `Two-stage + attn layers` attempts to prune heads and attention layers. We note that such approach leads to regression on MMLU and GSM8K.