Reasoning Models Can Be Accurately Pruned via Chain-of-Thought Reconstruction

Ryan Lucas[1,2], **Kayhan Behdin**[1], **Zhipeng Wang**[1], **Shao Tang**[1], **Qingquan Song**[1], **Rahul Mazumder**[1,2]
[1]LinkedIn, Sunnyvale, CA [2]Massachusetts Institute of Technology, Cambridge, MA

Abstract

Reasoning language models such as DeepSeek-R1 produce long chain-of-thought traces during inference time which make them costly to deploy at scale. Model pruning is a model compression technique that aims to reduce the model size by removing some weights from the model, while maintaining the accuracy in order to obtain more efficient models. We show that using compression techniques such as standard pruning methods produce large accuracy drop after pruning. To mitigate this, we introduce a simple, drop-in fix: during pruning we jointly reconstruct activations from the input and the model's own chain-of-thought traces via a layer-wise objective. This "Reasoning-Aware Compression" (RAC) integrates seamlessly into standard pruning methods such as SparseGPT, and boosts their performance significantly. We show that under this RAC approach, we can prune reasoning LLMs to 50% sparsity, while maintaining up to 95% of the original model's accuracy on math and coding tasks. Code reproducing the results in the paper can be found at: `https://github.com/RyanLucas3/RAC`

1 Introduction

Large Language Models (LLMs) with step-by-step reasoning abilities have become essential for solving complex, multi-step tasks in domains such as mathematics, coding, and logical reasoning [12]. Reasoning models generate explicit intermediate reasoning steps, or Chains-of-Thought (CoT) that significantly improve accuracy on challenging benchmarks, but at the cost of producing very long outputs for each query at inference time. For example, the DeepSeek-R1 model (671B parameters) achieves strong reasoning performance but must output lengthy explanation traces, making it extremely resource-intensive to deploy at scale [4, 13].

To reduce the cost of serving LLMs, recent years have seen a surge of interest in LLM compression techniques. Model pruning [5, 7] is a popular model compression technique where the goal is to remove redundant weights or neurons from the model, while ensuring the model quality remains high. This can lead to models with a smaller memory and compute footprint, which can be more resource efficient. Pruning has been particularly successful when applied to LLMs [3, 8, 11]. Given the computational requirement of reasoning LLMs, model pruning has appeared as an attractive proposition for efficient reasoning. However, the application of existing pruning methods to reasoning models can often result in significant accuracy loss [13], limiting the application of such compression techniques in practice where model quality is of high importance.

In this work, we focus on one-shot pruning of reasoning LLMs (that is, we do not conduct any retraining after pruning). We propose a new model pruning approach that better preserves the reasoning capabilities of LLMs. To this end, our pruning method aims to reduce the error arising from the model compression by minimizing the reconstruction error of model's *on-policy* CoT response, using a layer-wise objective function. This is in contrast to existing approaches such as [13] that do not make use of model's CoT when pruning. Our work demonstrates that by reconstructing the CoT, reasoning LLMs can be pruned to up to 50% sparsity in one-shot accurately, maintaining up to 95% of dense model's accuracy on math and coding tasks. Additionally, our proposed method

improves math and coding tasks' accuracy of pruned models by up to 17% compared to existing pruning approaches.

2 Background

Reasoning models. Conventional LLMs are trained to maximise the conditional likelihood $p_\theta(y_{0:L-1} \mid x) = \prod_{t=0}^{L-1} p_\theta\big(y_t \mid x, y_{<t}\big)$ of an output sequence $y_{0:L-1} \in \mathcal{V}^L$ of length L from the vocabulary $\mathcal{V}$ given a prompt x. A *reasoning model* instead produces:

$$(c_{0:T-1},\, y_{0:L-1}), \qquad \text{with } c_{0:T-1} \in \mathcal{V}^T,\ y_{0:L-1} \in \mathcal{V}^L,$$

where $c_{0:T-1}$ is a chain-of-thought (CoT) of length T and $y_{0:L-1}$ (hereafter abbreviated simply as y) is the final answer e.g. a single numeric value, a complete proof, or a code block.[1] While a conventional LLM can also be prompted to produce $c_{0:T-1}$ before its answer y, reasoning models are explicitly trained so that its generated chain $c_{0:T-1}$ outputs a verifiable task reward $R\big(x, c_{0:T-1}, y\big) \in [0, 1]$ for example, an exact match on a math problem, unit test pass for code, or logical-consistency checks for formal proofs. This has recently become popularized by the DEEPSEEK-R1 reasoning model, which is optimized via Group-Relative Policy Optimization (GRPO) [1].

LLM pruning methods. Modern language models contain billions of parameters, so *compression* is widely used for reducing GPU memory footprint, inference latency, and energy cost while preserving most of the model's accuracy. A popular approach to model compression is model pruning via a layer-wise objective [2]. Suppose a pretrained LLM with L layers is given, with layer weights $\mathbf{W}_\ell \in \mathbb{R}^{p_\ell \times d_\ell}$ for $\ell \in [L]$, and d_ℓ, p_ℓ denote the input and output sizes of layer ℓ, respectively. We also let $\mathbf{X}_\ell \in \mathbb{R}^{d_\ell \times N}$ denote the input activations to layer ℓ, gathered on a *calibration set* of N tokens. Layer-wise LLM pruning methods find compressed weights $\widehat{\mathbf{W}}_\ell$ by solving, independently for each layer,

$$\min_{\widehat{\mathbf{W}}_\ell} \big\|\mathbf{W}_\ell \mathbf{X}_\ell - \widehat{\mathbf{W}}_\ell \mathbf{X}_\ell\big\|_2^2 \ \text{ s.t. } \ \|\widehat{\boldsymbol{W}}\|_0 \le S \tag{1}$$

where $\|\cdot\|_0$ denotes the number of non-zero coordinates of a matrix and S is the desired number of non-zero coordinates. Numerous algorithms have been proposed for Layer-wise pruning of LLMs via solving (1) [3, 8, 11].

The calibration data. In equation 1, $\mathbf{X}_\ell$ is the so-called calibration data, and for LLMs a text corpus of size N tokens comprises the column dimension of $\mathbf{X}_\ell$. The calibration data is typically chosen to mimic the general distribution of natural language. As an example, the C4 dataset [9] is a common choice [3]. For standard LLMs, the calibration data is typically derived from a set of inputs (or prompts) x. Concretely, let $x_{0:N-1}$ be a batch of N prompt tokens from the calibration corpus and let $E \in \mathbb{R}^{d \times |\mathcal{V}|}$ denote the embedding matrix. Define the layer-wise hidden states for each token by:

$$\mathbf{x}_t^{(0)} \ = \ E\,\mathbf{e}_{x_t}, \qquad \mathbf{x}_t^{(j)} \ = \ f_j\big(\mathbf{x}_t^{(j-1)}\big),\ j = 1, \ldots, \ell - 1,$$

where f_j is the j^{th} transformer layer (including attention, MLP, residual connections, etc.), and $\mathbf{e}_{x_t}$ is the embedding for token t. The states are stacked column-wise to obtain the calibration activation matrix:

$$\mathbf{X}_\ell \ = \ \big[\,\mathbf{x}_0^{(\ell-1)},\ \mathbf{x}_1^{(\ell-1)},\ \ldots,\ \mathbf{x}_{N-1}^{(\ell-1)}\,\big] \in \mathbb{R}^{d_\ell \times N},$$

so the t-th token embedding (processed after $\ell - 1$ transformer layers) is exactly the vector that will enter layer ℓ when the dense model processes token x_t. This $\mathbf{X}_\ell$ is what standard pruning algorithms use to measure the reconstruction error of the compressed weight matrix $\widehat{\mathbf{W}}_\ell$. This makes sense, since in a standard LLM model we typically have $|x| \gg |y|$ (long context, short reply). However, for reasoning LMs we observe the opposite regime $|c_{0:T-1}| + |y| \gg |x|$. The model's CoT will typically be much longer than the input question (e.g., a math problem).

Pruning of reasoning LLMs Concurrently with our work, Zhang et al. [13] run an extensive evaluation of compressed DeepSeek-R1 variants. They apply SparseGPT [3] to student models distilled from Qwen and LLaMA, but follow the default C4-style calibration pipeline (details of the calibration set are not reported). Their results are similar to what we observe: when using a generic

[1] See Wei et al. [12] for evidence that even noisy CoT traces boost reasoning accuracy.

dataset, accuracy on complex reasoning tasks drops sharply with sparsity, chains of thought become repetitive or degenerate, and longer post-compression outputs correlate with lower task accuracy. Unlike their empirical study, our reasoning-aware method modifies the calibration distribution itself injecting on-policy CoT activations, and thereby mitigates the performance loss during pruning.

3 Reasoning-Aware Compression (RAC)

Inference in an autoregressive reasoning model. Let $x = (x_0, \ldots, x_{T_{\text{in}}-1}) \in \mathcal{V}^{T_{\text{in}}}$ be a prompt to an autoregressive reasoning model, and let π_θ denote the dense model's distribution over the vocabulary $\mathcal{V}$. At inference the model generates a full sequence:

$$z_{0:T} = \big(x_0, \ldots, x_{T_{\text{in}}-1},\, c_{T_{\text{in}}}, \ldots, c_T\big),$$

where $\mathcal{P} = \{0, \ldots, T_{\text{in}} - 1\}$ indexes prompt tokens and $\mathcal{D} = \{T_{\text{in}}, \ldots, T\}$ indexes the decode tokens. Each decode token c_t is drawn autoregressively:

$$c_t \sim \pi_\theta(\cdot \mid z_{0:t-1}), \qquad t \in \mathcal{D}.$$

At each step $t \in \mathcal{P} \cup \mathcal{D}$ the model computes hidden states:

$$\mathbf{x}_t^{(0)} = E\, e_{z_t}, \qquad \mathbf{x}_t^{(\ell)} = f_\ell\Big(\{\mathbf{x}_\tau^{(\ell-1)}\}_{\tau \le t}\Big), \quad \ell = 1, \ldots, L,$$

That is, crucially, to generated the complete sequence, the model relies on the activations that are computed on the input, but also on activations that arise from its own self-generated tokens. Once the final hidden state $\mathbf{x}_t^{(L)}$ is computed, it is mapped to vocabulary logits via the output projection $W_{\text{out}} \in \mathbb{R}^{|\mathcal{V}| \times d_L}$ given by $\boldsymbol{y}_t = W_{\text{out}}\, \mathbf{x}_t^{(L)} \in \mathbb{R}^{|\mathcal{V}|}$. which gives the next token distribution $\pi_\theta(\cdot \mid z_{0:t}) = \text{softmax}(\boldsymbol{y}_t)$.

Aligning compression with decoding during offline calibration. Suppose we have M calibration prompts $\{x^{(m)}\}_{m=1}^M$, with corresponding prompt index sets $\mathcal{P}_m$ and decode index sets $\mathcal{D}_m$. Standard post-training pruning collects activations only for $t \in \mathcal{P}_m$, i.e. from fixed prompt tokens. In reasoning tasks, however, $|\mathcal{D}_m| \gg |\mathcal{P}_m|$, so ignoring decode-time activations misses the majority of inference computation. RAC modifies calibration by self-generating tokens during calibration to simulate decode activations. At each step $t \in \mathcal{D}_m$, the model's own prediction is re-used as the next input:

$$z_{t+1}^{(m)} \sim \pi_\theta(\cdot \mid z_{0:t}^{(m)}), \qquad \pi_\theta(\cdot \mid z_{0:t}^{(m)}) = \text{softmax}\big(W_{\text{out}}\, \mathbf{x}_t^{(L,m)}\big).$$

The embedding and hidden states for this token are then computed:

$$\mathbf{x}_{t+1}^{(0,m)} = E\, e_{z_{t+1}^{(m)}}, \qquad \mathbf{x}_{t+1}^{(\ell,m)} = f_\ell\Big(\{\mathbf{x}_\tau^{(\ell-1,m)}\}_{\tau \le t+1}\Big),\ \ell = 1, \ldots, L.$$

The resulting layer-ℓ inputs are appended to the decode activation matrix $\mathbf{X}_\ell^{\text{D}} \leftarrow \big[\, \mathbf{X}_\ell^{\text{D}} \;\; \mathbf{x}_{t+1}^{(\ell-1,m)} \,\big]$. After all steps, the final calibration matrix concatenates prompt and decode activations $\mathbf{X}_\ell^{\text{RAC}} = \big[\, \mathbf{X}_\ell^{\text{P}} \;\; \mathbf{X}_\ell^{\text{D}} \,\big] \in \mathbb{R}^{d_\ell \times (N_{\text{P}} + N_{\text{D}})}$. Pruning is then performed by minimizing, for each layer ℓ,

$$\|(\mathbf{W}_\ell - \widehat{\mathbf{W}}_\ell)\, \mathbf{X}_\ell^{\text{RAC}}\|_F^2 = \sum_{m=1}^{M} \sum_{t \in \mathcal{P}_m \cup \mathcal{D}_m} \|(\mathbf{W}_\ell - \widehat{\mathbf{W}}_\ell)\, \mathbf{x}_t^{(\ell-1,m)}\|_2^2.$$

4 Experiments

Experimental Setup. To test the effectiveness of reasoning-aware compression, we perform one-shot pruning on several open-source Qwen architectures (1.5B, 7B, 14B) which have been distilled from the DeepSeek-R1 model. Each model is pruned in one-shot with SparseGPT at layer-wise unstructured sparsity of 20%, 30%, 40% and 50% using 1M calibration tokens from: (i) the standard English–web C4 corpus, (ii) OPEN-R1-MATH-220K [6] or CODEFORCES [10] problem statements without answers or reasoning traces ("prompt only") and (iii) those prompts augmented with up to $T_{\text{max}} = 8192$ on–policy CoT tokens collected from each corresponding dense model ("RAC"). For all three calibration settings (C4, prompt-only, and RAC), we fix the

calibration budget to 1M tokens, ensuring a fair comparison that avoids the confounder of RAC simply benefiting from more tokens due to decoding.

For mathematical reasoning we use MATH500 and report **acc@1:1**: the percentage of problems for which the model's single most-confident prediction exactly matches the ground-truth answer (Top-1 accuracy). For code generation we use the CodeGen evaluation harness and report **acc@1:16**, i.e., the percentage of cases in which the correct solution appears anywhere within the model's top-16 predictions, irrespective of rank (Top-16 accuracy). All evaluations are zero-shot with no additional few-shot examples, and a 32k output token budget. This closely follows the evaluation pipeline used by DeepSeek [1] and the open-source replication of the DeepSeek pipeline from [6].

Table 1: Comparison of pruning approaches when applied to DeepSeek-R1 distilled models, across various sparsity levels. Accuracy with standard error (SE). Best accuracy per row in **green**.

		Math500 acc@1:1			*codegen_pass@1:16*		
Model	Sparsity	C4	Prompt only	RAC	C4	Prompt only	RAC
1.5B	*Dense*	**0.832**	**0.832**	**0.832**	**0.161**	**0.161**	**0.161**
	20%	0.822 (0.017)	**0.840 (0.016)**	0.832 (0.017)	0.148 (0.018)	**0.156 (0.019)**	0.155 (0.018)
	30%	0.762 (0.019)	0.788 (0.018)	**0.822 (0.017)**	0.127 (0.017)	0.138 (0.018)	**0.150 (0.018)**
	40%	0.658 (0.021)	0.728 (0.020)	**0.774 (0.019)**	0.066 (0.011)	0.086 (0.013)	**0.129 (0.017)**
	50%	0.356 (0.021)	0.496 (0.022)	**0.664 (0.021)**	0.004 (0.002)	0.024 (0.006)	**0.093 (0.014)**
7B	*Dense*	**0.936**	**0.936**	**0.936**	**0.374**	**0.374**	**0.374**
	20%	0.902 (0.013)	0.928 (0.012)	**0.934 (0.011)**	0.362 (0.025)	**0.367 (0.025)**	0.364 (0.025)
	30%	0.904 (0.013)	0.922 (0.012)	**0.934 (0.011)**	0.335 (0.025)	0.341 (0.025)	**0.361 (0.025)**
	40%	0.890 (0.014)	0.898 (0.014)	**0.912 (0.013)**	0.273 (0.023)	0.300 (0.024)	**0.333 (0.024)**
	50%	0.744 (0.020)	0.812 (0.017)	**0.900 (0.013)**	0.099 (0.014)	0.228 (0.021)	**0.283 (0.023)**
14B	*Dense*	**0.941**	**0.941**	**0.941**	**0.513**	**0.513**	**0.513**
	20%	0.952 (0.010)	0.954 (0.009)	**0.962 (0.009)**	0.508 (0.027)	0.508 (0.027)	0.508 (0.027)
	30%	0.936 (0.011)	0.930 (0.011)	**0.936 (0.011)**	0.491 (0.027)	**0.496 (0.027)**	**0.496 (0.027)**
	40%	0.910 (0.013)	0.928 (0.012)	**0.942 (0.010)**	0.447 (0.026)	0.471 (0.027)	**0.480 (0.027)**
	50%	0.878 (0.015)	0.880 (0.015)	**0.910 (0.013)**	0.319 (0.024)	0.385 (0.026)	**0.424 (0.026)**

Discussion of pruning results. Table 1 presents the accuracy@1:1 results (exact match accuracy) for mathematical reasoning tasks on the Math500 benchmark across different model sizes and sparsity levels. The results demonstrate that RAC consistently outperforms both baseline calibration methods, particularly at higher sparsity levels where standard compression techniques cause severe performance loss. At 50% sparsity on the 1.5B parameter model, RAC achieves 66.4% accuracy compared to only 35.6% with standard C4 calibration. Similarly, for the 7B model at 50% sparsity, RAC maintains 90.0% accuracy while C4 calibration drops to 74.4%. The improvements are less pronounced at lower sparsity levels, with all methods performing similarly to the dense model at 20% sparsity, suggesting that the benefits of reasoning-aware calibration become more pronounced as compression becomes more aggressive.

Additionally, Table 1 shows the pass@1:16 results for code generation tasks on LiveCodeBench, where the metric represents whether the correct solution appears anywhere within the model's top-16 predictions. The pattern observed here mirrors the mathematical reasoning results, with RAC providing the largest performance gains at high sparsity levels. However, code generation tasks appear to be even more sensitive to compression than mathematical reasoning. Across both tasks, the effectiveness of RAC scales directly with sparsity, becoming more beneficial as pruning becomes more aggressive. Even the approach of using task-specific prompts without chain-of-thought traces (prompt-only calibration) consistently outperforms generic C4 calibration, indicating that domain-relevant calibration data matters significantly for reasoning tasks. However, the full RAC approach that incorporates complete reasoning traces provides additional substantial benefits beyond task-specific prompts alone. Finally, larger models demonstrate greater robustness to compression across all methods, with the 14B model maintaining higher performance at equivalent sparsity levels compared to the smaller variants, though RAC still provides meaningful improvements for these models.

References

[1] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

[2] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35: 4475–4488, 2022.

[3] Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 10323–10337. PMLR, 2023.

[4] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[5] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[6] HuggingFace. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

[7] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[8] Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. Alps: Improved optimization for highly sparse one-shot pruning for large language models, 2024. URL `https://arxiv.org/abs/2406.07831`.

[9] 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.

[10] Jiaxi Yang Bowen Yu Bo Zheng Dayiheng Liu Shanghaoran Quan. Codeforces: Benchmarking competition-level code generation of llms on codeforces. 2025. Disclaimer: This is a non-traditional code benchmark.

[11] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models, 2024. URL https://arxiv.org/abs/2306.11695.

[12] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837, 2022.

[13] Nan Zhang, Yusen Zhang, Prasenjit Mitra, and Rui Zhang. When reasoning meets compression: Benchmarking compressed large reasoning models on complex reasoning tasks. *arXiv preprint arXiv:2504.02010*, 2025. URL https://arxiv.org/abs/2504.02010.

5 Acknowledgement

Ryan Lucas contributed to this work while he was an intern at LinkedIn during summer 2025. This work is not a part of his MIT research. Rahul Mazumder contributed to this work while he was a consultant for LinkedIn (in compliance with MIT's outside professional activities policies). This work is not a part of his MIT research.