Winning at the RL Casino

Scott Biggs
Khoury College of Computer Science
Northeastern University
San Jose, CA 95112
biggs.s@northeastern.edu

Abhinav Khushalani
Khoury College of Computer Science
Northeastern University
San Jose, CA 95112
khushalani.a@northeastern.edu

Abstract

Reinforcement Learning (RL) is a widely used in LLMs, and has been broadly successful post-training and domain fine-tuning. However, RL is notoriously unstable, environment sensitive, and compute hungry. In this work, we lay a serious foundation for a new class of RL methods that leverage the empirical sparsity of weight-updates in LLMs undergoing RL training. Our method, ***BSR-AdamW***, demonstrates a persistent $\sim 10\%$ reduction in training times vs the baseline implementations of AdamW in PyTorch through HuggingFace and Transformer Reinforcement Learning (TRL) without compromising the quality of trained models. ***BSR-AdamW***is a serious application of recent work from across the Graph-ML, Deep Weight Space, and RL/LLM communities to produce concrete optimizations for RL in LLMs.

1 Introduction

1.1 The Computational Burden of Reinforcement Learning

Reinforcement Learning has become indispensable for post-training large language models, enabling alignment with human preferences and complex reasoning capabilities. From DeepMind's AlphaGo to modern RLHF systems that align billion-parameter LLMs, RL has proven its value across domains [DeepSeek-AI et al., 2025, Silver et al., 2017]. Yet this success masks a fundamental challenge: computational cost. Training state-of-the-art RL agents demands massive resources, with estimated cloud compute bills reaching $12-18 million for flagship systems, placing cutting-edge RL research beyond the reach of most academic and industry teams.

The core assumption underlying this computational burden is deceptively simple: meaningful learning requires updating most or all model parameters. For LLMs undergoing RL fine-tuning—whether through DPO, PPO, or related methods—this translates to billions of parameter updates per step, each demanding forward passes, gradient computation, and optimizer state management. As models scale from millions to hundreds of billions of parameters, this assumption becomes increasingly expensive to maintain DeepSeek-AI et al. [2025], Rafailov et al. [2024].

1.2 An Empirical Basis for Sparsity

But what if this assumption is wrong? Recent work by Mukherjee et al. [2025] suggests exactly that. In a comprehensive study across multiple RL algorithms and model families, they observed that RL fine-tuning consistently modifies only a small subnetwork—typically $5 - 30\%$ of weights—leaving the vast majority of parameters unchanged [Liu et al., 2021, Mukherjee et al., 2025]. This "RL-induced parameter update sparsity" emerges naturally, without any explicit sparsity constraints, parameter-efficient tuning methods, or architectural modifications. The subnetworks updated by

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

RL show substantial overlap across different random seeds, datasets, and algorithms, suggesting a partially transferable structure inherent to the pretrained model.

This observation resonates with the lottery ticket hypothesis Chen et al. [2020], Frankle and Carbin [2019]: dense networks contain sparse subnetworks that, when trained in isolation, can match full network performance. However, RL training presents a fascinating inversion of this principle. Rather than searching for winning tickets that can train from scratch, RL appears to naturally select and modify a pre-existing subnetwork, leaving a "lottery ticket" that was effectively pre-drawn by pretraining. This may explain a long-standing puzzle: why does RLHF preserve pretrained capabilities better than supervised fine-tuning? The answer may be remarkably simple — it leaves most weights, and thus most 'knowledge', untouched.

1.3 Our Contribution

The empirical sparsity of RL updates is intellectually compelling, but its practical implications remain unexplored. If RL training naturally concentrates on sparse subnetworks, can we identify these regions early and restrict computation accordingly? Can we achieve true wall-clock speedups by avoiding computation on inactive parameters, rather than merely reducing memory footprint? And crucially, can we do this without sacrificing model quality?

This paper presents ***BSR-AdamW***, a rigorous framework for exploiting RL-induced sparsity to accelerate training through kernel level manipulation of the optimizer. We demonstrate that the sparse subnetworks where RL operates can be identified reliably from early training dynamics, and that restricting training to these subnetworks yields substantial computational savings. Our contributions are threefold:

1.4 Principled Subnetwork Discovery

We develop three complementary masking methods operating on different principles: magnitude-based accumulation (parameters with largest cumulative changes), momentum-based selection (parameters with consistent, directional updates), and Fisher information approximation (parameters with high variance and magnitude). Each method processes streaming weight deltas from early checkpoints, enabling prediction of which parameters will be active throughout training. We introduce an exact top-k mask construction algorithm with global thresholding and tie-breaking that reliably achieves target sparsity levels within tolerable error.

1.4.1 Hardware-Accelerated Sparse Optimization

We introduce ***BSR-AdamW***, a Triton-accelerated sparse optimizer leveraging Block Sparse Row representations. Unlike parameter-efficient fine-tuning methods that reduce memory but still compute gradients for all parameters, ***BSR-AdamW***processes only non-zero mask elements through indexed gather-scatter operations. This yields true computational acceleration—we achieve persistent $\sim 10\%$ speedups in wall-clock training time across model scales and RL algorithms [Kingma and Ba, 2017, Loshchilov and Hutter, 2019].

1.4.2 Quality-Preserving Training at High Sparsity

We validate that training with only 2.5% of weight changes can match or exceed dense training performance on standard benchmarks (MATH-500, GPQA-Diamond, MMLU) while maintaining the speedup benefits. Through Jaccard similarity analysis, we quantify mask prediction quality and demonstrate that our methods capture the true subnetworks of RL training with high fidelity section 3.

1.5 Related Literature

Our work bridges several active research areas while offering distinct advantages. The lottery ticket hypothesis (LTH) demonstrated that dense neural networks contain sparse subnetworks that, when trained in isolation, can match or exceed the performance of the full model [Frankle and Carbin, 2019]. Subsequent work showed that such winning tickets can be identified early in training [Frankle et al., 2021], generalized across optimizers and initializations [Morcos et al., 2019], and extended to large-scale architectures including transformers [Chen et al., 2020, Zheng et al., 2022]. These findings motivated a broad line of research into sparse training as a means of reducing both memory and compute requirements without sacrificing accuracy.

Dynamic sparse training (DST) methods aim to maintain sparsity throughout training by periodically updating connectivity patterns. RigL [Evci et al., 2020] showed that gradient-based mask updates allow sparse networks to match dense baselines while avoiding iterative prune–retrain cycles. Follow-up work explored alternative growth criteria [Mostafa and Wang, 2019], stability properties of sparse masks [Liu et al., 2021], and the behavior of sparse connectivity in large-scale models [Evci et al., 2021]. However, these approaches typically assume training from scratch and introduce nontrivial overhead from mask updates, gradient tracking, or rewiring heuristics. In contrast, we demonstrate that reinforcement learning (RL) fine-tuning naturally induces a stable sparse structure, which can be directly exploited without explicit sparsity-inducing objectives or iterative pruning schedules.

Parameter-efficient fine-tuning (PEFT) methods—including adapters [Houlsby et al., 2019], prefix tuning [Li and Liang, 2021], and LoRA [Hu et al., 2022]—have become standard practice for adapting large language models. These approaches significantly reduce the number of trainable parameters and optimizer state, enabling fine-tuning on limited hardware. However, PEFT methods fundamentally differ from our approach: they retain dense forward and backward passes through the base model and thus do not eliminate the majority of compute. As a result, PEFT primarily yields memory savings rather than true wall-clock acceleration. Unlike LoRA, which reduces trainable parameters but still computes dense gradients, ***BSR-AdamW*** eliminates computation on inactive weights entirely.

Our work is also related to research on reinforcement learning optimization and scaling. Prior efforts to accelerate RL training have focused primarily on increasing environment throughput via parallel simulation [Mnih et al., 2016, Espeholt et al., 2018] or leveraging large-scale distributed systems for complex environments [Berner et al., 2019, Vinyals et al., 2019]. While these methods improve sample efficiency and utilization, they do not address the growing cost of parameter updates as model sizes increase. Recent large-scale RLHF pipelines for language models further exacerbate this imbalance, as optimizer and backpropagation costs dominate runtime [Ouyang et al., 2022, Bai et al., 2022]. Our approach is complementary: by sparsifying the update step itself, we directly reduce the per-iteration computational burden of RL fine-tuning.

Finally, structured sparsity and hardware-aware pruning have demonstrated the potential for end-to-end acceleration when sparsity aligns with accelerator primitives. N:M sparsity patterns enable efficient execution on modern GPUs via tensor cores [NVIDIA, 2020, Zhang et al., 2023], while block-structured pruning reduces memory fragmentation and improves cache efficiency [Wen et al., 2016, Gray et al., 2017]. Although such methods can offer substantial speedups, they often impose rigid constraints on sparsity patterns or require retraining to maintain accuracy. Our block sparse row (BSR) representation provides a flexible alternative that supports arbitrary sparsity levels and adapts naturally to the sparsity emerging from RL dynamics, while remaining compatible with future hardware-specific optimizations.

In summary, while prior work has explored sparse training, PEFT, and RL acceleration largely in isolation, ***BSR-AdamW*** unifies these directions by showing that reinforcement learning fine-tuning intrinsically reveals sparse subnetworks that can be exploited for real computational gains—without retraining, auxiliary sparsity objectives, or restrictive structural assumptions.

2 Methods

2.1 Subnetwork Mask Finding

In order to implement the planned sparse optimizations with Triton, we must first identify the subnetwork to target. We present three basic approaches and their motivations below.

Table 1: Comparison of Sparse Mask Generation Methods for RL Fine-Tuning

Method	Intuition	Score Function	Memory Complexity
Magnitude Masking	Parameters accumulating large changes are consistently important	$s_i = \sum_{t=1}^{T} \lvert\delta_i^t\rvert$	$O(\lvert\theta\rvert)$ Streaming accumulation
Momentum Masking	Consistent, large velocity indicates persistent gradient flow	$s_i = \frac{\lvert\bar{v}_i\rvert^2}{\sigma_{v_i} + \epsilon}, \quad v_i^t = \delta_i^t - \delta_i^{t-1}$	$O(w \cdot \lvert\theta\rvert)$ Sliding window of w velocities
Fisher Approximation	High variance + magnitude indicates parameter sensitivity	$\mathcal{F}_i \approx \mathrm{Var}[\delta_i] + \lvert\mathbb{E}[\delta_i]\rvert$	$O(\lvert\theta\rvert)$ Welford's online variance

Notation: δ_i^t is the weight change at step t for parameter i; $|\theta|$ is total parameter count; T is number of training steps; w is momentum window size. All methods use GPU-accelerated streaming to avoid loading all checkpoints into memory simultaneously.

More concretely, the motivations for these mask finding approaches can be summarized in the context of the optimization process as follows:
* Magnitude: Greater cumulative SGD path length
* Momentum: Persistent gradient alignment across steps
* Fisher approx: local curvature / sensitivity in loss space

These procedures allow us to generate a dictionary $S = (n_i, m_i)$ of 'score' criteria from which to build a binary mask $M = (n_i, m_i) \in \{0, 1\}$ of included and excluded parameters.

2.2 Subnetwork Mask Construction

1 presents a pseudocode implementation of this mask construction algorithm. Two key features are 1.) light noise addition, to break ties in S, and 2.) global threshold estimation, which estimates a global S threshold score of parameter indicies to be included in the mask. We find that these global threshold estimations typically align with graph-based subnetwork identification approaches [CITATION].

Algorithm 1: Approximate $\rho\%$ Sparsity Mask Creation with Global Threshold

Input: Score dictionary $S = \{(n_i, s_i)\}$, target sparsity $\rho \in [0, 100]$, device d
Output: Binary mask dictionary $M = \{(n_i, m_i)\}$ where $m_{ij} \in \{0, 1\}$

$k_{\text{pct}} \leftarrow 100 - \rho$ `// Keep percentage`
`// Step 1: Tie-breaking via noise injection`
foreach $(name, score) \in S$ **do**
    $\varepsilon \leftarrow \max(\|score\|_\infty \times 10^{-10}, 10^{-12})$;
    $noise \sim \mathcal{N}(0, \varepsilon^2)$ `// Gaussian noise`
    $S[name] \leftarrow score + noise$;
`// Step 2: Estimate global threshold via sampling`
$samples \leftarrow []$;
$N_{total} \leftarrow 0$;
foreach $(name, score) \in S$ **do**
    $N_{total} \leftarrow N_{total} + |score|$ `// Parameter count`
    $flat \leftarrow \text{flatten}(score)$;
    $n_{sample} \leftarrow \min(100000, |flat|)$;
    $indices \leftarrow \text{random\_permutation}(|flat|)[1 : n_{sample}]$;
    $samples.\text{append}(flat[indices])$;
$all\_samples \leftarrow \text{concatenate}(samples).\text{to}(d)$;
$k_{target} \leftarrow \max(1, \lfloor k_{\text{pct}}/100 \times N_{total} \rfloor)$;
$k_{sample} \leftarrow \max(1, \lfloor k_{\text{pct}}/100 \times |all\_samples| \rfloor)$;
if $k_{sample} \geq |all\_samples|$ **then**
    $\theta_{global} \leftarrow 0$;
else
    $top_k \leftarrow \text{TopK}(all\_samples, k_{sample})$;
    $\theta_{global} \leftarrow \min(top_k)$;
`// Step 3: Apply threshold per-layer`
$M \leftarrow \{\}$;
$N_{kept} \leftarrow 0$;
foreach $(name, score) \in S$ **do**
    $M[name] \leftarrow \mathbb{1}_{score \geq \theta_{global}}$;
    $N_{kept} \leftarrow N_{kept} + \sum M[name]$;
$\rho_{actual} \leftarrow 100 \times (1 - N_{kept}/N_{total})$;
`// Step 4: Correction if error` $> 5\%$
if $|\rho_{actual} - \rho| > 5$ **then**
    $M \leftarrow \{\}$;
    $N_{kept} \leftarrow 0$;
    foreach $(name, score) \in S$ **do**
        $score_{gpu} \leftarrow score.\text{to}(d)$;
        $flat \leftarrow \text{flatten}(score_{gpu})$;
        $k_{layer} \leftarrow \max(1, \lfloor k_{\text{pct}}/100 \times |flat| \rfloor)$;
        if $k_{layer} \geq |flat|$ **then**
            $M[name] \leftarrow \mathbf{1}_{score_{gpu}}$ `// All ones`
        else
            $top_k \leftarrow \text{TopK}(flat, k_{layer})$;
            $\theta_{local} \leftarrow \min(top_k)$;
            $M[name] \leftarrow \mathbb{1}_{score_{gpu} \geq \theta_{local}}$;
        $N_{kept} \leftarrow N_{kept} + \sum M[name]$;
return M;

2.3 AdamW with Block Sparse Representations

BSR-AdamW leverages a BSR optimized Triton GPU kernel to reduce the memory load of the AdamW optimizer [Kingma and Ba, 2017, Loshchilov and Hutter, 2019, Wen et al., 2025b]. We use the binary mask M found by 1 to find a BSR representation of the subnetwork weights and weight moments calculated. As a result, ***BSR-AdamW***has time complexity $\mathcal{O}(N(1-\rho))$, for a sparsity coefficient ρ, as opposed to the baseline AdamW complexity $\mathcal{O}(N)$.

Algorithm 2: Indexed Sparse AdamW Step

Input: Parameters θ, gradients g, non-zero indices $\mathcal{I} = \{i : M_i = 1\}$, momentum states m_t, v_t, hyperparameters $\alpha, \beta_1, \beta_2, \epsilon, \lambda$, step t

Output: Updated parameters θ_{t+1} and states m_{t+1}, v_{t+1}

`// Precompute bias corrections on CPU (avoids kernel recompilation)`

$\hat{\beta}_1 \leftarrow 1 - \beta_1^t$;

$\hat{\beta}_2 \leftarrow 1 - \beta_2^t$;

`// GPU Kernel: Process only non-zero indices in parallel`

foreach $i \in \mathcal{I}$; `// parallel execution`

do

> `// Gather: Load only active parameters`
>
> $\theta_i, g_i, m_{t,i}, v_{t,i} \leftarrow \text{load}(\theta[i], g[i], m_t[i], v_t[i])$;
>
> `// Weight decay (decoupled)`
>
> $\theta_i \leftarrow \theta_i \cdot (1 - \alpha\lambda)$;
>
> `// Update biased moments`
>
> $m_{t+1,i} \leftarrow \beta_1 m_{t,i} + (1-\beta_1) g_i$;
>
> $v_{t+1,i} \leftarrow \beta_2 v_{t,i} + (1-\beta_2) g_i^2$;
>
> `// Bias-corrected moments`
>
> $\hat{m}_i \leftarrow m_{t+1,i} / \hat{\beta}_1$;
>
> $\hat{v}_i \leftarrow v_{t+1,i} / \hat{\beta}_2$;
>
> `// Parameter update`
>
> $\theta_{t+1,i} \leftarrow \theta_i - \alpha \cdot \frac{\hat{m}_i}{\sqrt{\hat{v}_i} + \epsilon}$;
>
> `// Scatter: Store only to active locations`
>
> $\text{store}(\theta_{t+1,i}, m_{t+1,i}, v_{t+1,i})$;

Key optimizations:

- **Indexed operations:** Only $|\mathcal{I}| = (1-\rho) \cdot |\theta|$ parameters are accessed, achieving $O(|\mathcal{I}|)$ complexity vs. $O(|\theta|)$ for dense AdamW.
- **Bias correction precomputation:** Computing $1 - \beta^t$ on CPU (microseconds) avoids expensive GPU power operations and kernel recompilation overhead (50-200ms per step).
- **Memory coalescing:** Triton's block-based parallelism ensures efficient GPU memory access patterns through vectorized loads/stores.

Typically, the mask would be found from the early 'warm start' weight deltas logged from a dense run, then would be applied with ***BSR-AdamW***to accelerate the remainder of the run from that checkpoint.

3 Results

All experiments were performed on a single NVIDIA H200 GPU.

3.1 Mask Quality

To measure the quality of the masks our methods uncover from weight-delta checkpoints, we overfit Gemma 3 270M Instruct for 1000 steps on the Light-R1 DPO fine tuning dataset [Team et al., 2025, Wen et al., 2025b, Rafailov et al., 2024].

By studying the 'true' deltas from across the run, we can quantify the quality of our reconstructions with a Jaccard score between the true mask M_* and prediction M_{pred} in weight-index space at the target sparsity.

$$J(M_{pred}, M_*) = \frac{|M_{pred} \cap M_*|}{|M_{pred} \cup M_*|} \in [0, 1]$$

Predicted masks are built from only the first 100 steps logged, whereas the true masks are the absolute magnitude delta results from across the entire run. The results of this ablation are presented in table Table 2.

Future enhancements to the mask evaluation criterion would be helpful, particularly with respect to identifying the 'true' mask Evci et al. [2020].

Table 2: Jaccard Similarity of Mask Prediction Methods Across Sparsity Levels (Gemma 3 270M)

Sparsity	Momentum (window 10)	Magnitude	Fisher
90%	0.1125	0.2754	**0.2761**
95%	0.1734	**0.7713**	0.7455
97.5%	**0.6644**	0.6642	0.658
99%	**0.4248**	0.3516	0.3575

Jaccard similarity measures agreement between predicted masks and ground truth (final checkpoint). Higher values indicate better early prediction. Bold indicates best method per sparsity level.

3.2 Timing Accelerations

Training step timing evaluations were repeated several times, and ***BSR-AdamW***consistently showed a 10-15% reduction in training time against the baseline. However, the absolute values of the timings across GPU sessions varied significantly. While we did not control for this hardware 'warm up', we observed persistent gains regardless of variable absolute timing values across runs. Timing measurements were taking during complete steps from the optimizers point of view, including forward and backward passes through the LLM. All experiments, including timing measures, were taken on a single NVIDIA H200 GPU.

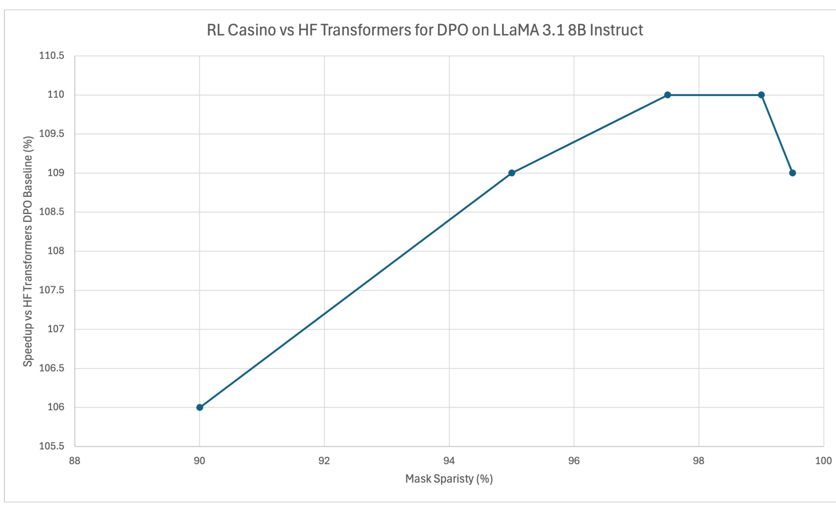

Figure 1: Relative acceleration of ***BSR-AdamW***vs the Transformers AdamW implementation [Kingma and Ba, 2017, Loshchilov and Hutter, 2019]. Acceleration is calculated as $t_{\text{AdamW}}/t_{\textbf{\textit{BSR-AdamW}}} \times 100\%$. Higher values indicate superior relative performance.

3.3 LLM Evaluations

We evaluate LLaMA 3.1 8B Instruct [Grattafiori et al., 2024] tuned with DPO for 100 steps on a subset of 1000 samples from the Light-R1 dataset [Wen et al., 2025a], pulled from HuggingFace, on three key benchmarks: MATH-500 (or Hendricks-500), GPQA-Diamond, and MMLU [Hendrycks et al., 2021, Rein et al., 2023]. The mask used in Figure 1 had $\sim 97.5\%$ sparsity, and calculated from weight delta logs between step 10 and 40 in a warm start dense DPO run. The mask was applied to the base unmodified model for the purposes of fairness in the comparison, although in practice it would be applied to the same model from which it was calculated and training would continue from that checkpoint.

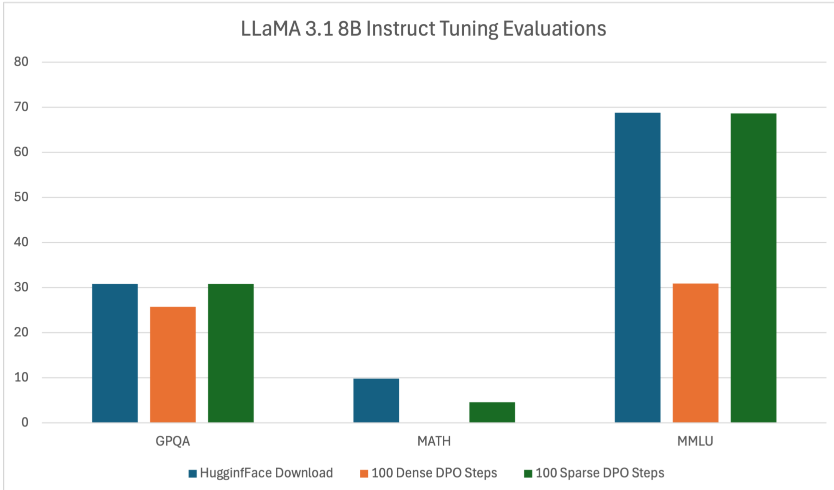

Figure 2: Comparison of LLaMA 3.1 8B Instruct models pulled from HuggingFace before and after sparse ***BSR-AdamW*** training and typical dense training.

Despite probable issues with the lm-eval harness used, particularly with the MATH benchmark, models trained with ***BSR-AdamW*** equal or outperform dense training in all settings.

4 Conclusion

BSR-AdamW represents an important first step toward efficient RL training through sparsity exploitation, but many exciting directions remain open. Dynamic mask updates using RigL-style approaches could adapt to changing training dynamics [Evci et al., 2020]. Sparse-aware learning rate schedules might better account for the reduced effective parameter count Zhang et al. [2023], Mukherjee et al. [2025]. Hierarchical sparsity patterns could exploit structure at multiple granularities. Extensions beyond DPO and GRPO to other RL algorithms—PPO, REINFORCE, actor-critic methods—would broaden applicability Mukherjee et al. [2025], DeepSeek-AI et al. [2025]. Investigation of when and why certain parameters become active could yield deeper insights into RL training dynamics.

There are more supplementary experiments to run on ***BSR-AdamW*** that we did not have the time or resources to complete, including ablations over the sparsity/speed/quality tradeoff, comparison against LoRA PEFT to examine the 'full rank' nature of RL updates, and comaprison to graph pruning based 'true' subnetwork identification methods Chen et al. [2020], Evci et al. [2020], Mukherjee et al. [2025].

By establishing principled methods for identifying and exploiting the natural sparsity of RL training, we make high-quality reinforcement learning more computationally accessible. In the metaphorical "RL Casino" where training happens, understanding precisely where learning concentrates—and betting only on those locations—is how you win.

5 Works Cited

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Zachary Goldstein, Stanislav Fort, Sam McCandlish, et al. Constitutional AI: Harmlessness from AI feedback, 2022. URL `https://arxiv.org/abs/2212.08073`.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 15834–15846, 2020.

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`.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL `https://arxiv.org/abs/1802.01561`.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. *International Conference on Machine Learning*, pages 2943–2952, 2020.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2021. URL `https://arxiv.org/abs/1911.11134`.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL `https://arxiv.org/abs/1803.03635`.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=Ig-VyQc-MLK`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego

Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim

Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Scott Gray, Alec Radford, and Diederik P. Kingma. Gpu kernels for block-sparse weights, 2017. URL `https://cdn.openai.com/blocksparse/blocksparsepaper.pdf`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL `https://arxiv.org/abs/2009.03300`.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL `https://arxiv.org/abs/1412.6980`.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597. Association for Computational Linguistics, 2021. URL `https://aclanthology.org/2021.acl-long.353/`.

Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware, 2021. URL `https://arxiv.org/abs/1901.09181`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL `https://arxiv.org/abs/1711.05101`.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. URL `https://arxiv.org/abs/1602.01783`.

Ari S. Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: Generalizing lottery ticket initializations across datasets and optimizers, 2019. URL `https://arxiv.org/abs/1906.02773`.

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning (ICML)*, pages 4646–4655. PMLR, 2019. URL `https://proceedings.mlr.press/v97/mostafa19a.html`.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tür, and Hao Peng. Reinforcement learning fine-tunes small subnetworks in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=0NdS4xCngO`.

NVIDIA. Nvidia A100 tensor core GPU architecture. Whitepaper, 2020. URL `https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL `https://arxiv.org/abs/2305.18290`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL `https://arxiv.org/abs/2311.12022`.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL `https://arxiv.org/abs/2503.19786`.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. volume 575, pages 350–354. Nature Publishing Group, 2019.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*, 2025a.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond, 2025b. URL `https://arxiv.org/abs/2503.10460`.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks, 2016. URL `https://arxiv.org/abs/1608.03665`.

Yuxin Zhang, Yiting Luo, Mingbao Lin, Yunshan Zhong, Jingjing Xie, Fei Chao, and Rongrong Ji. Bi-directional masks for efficient n:m sparse training, 2023. URL `https://arxiv.org/abs/2302.06058`.

Rui Zheng, Junjie Wen, Di He, Yelong Shen, Tie-Yan Liu, and Weizhu Chen. Robust lottery tickets for pre-trained language models, 2022. URL `https://arxiv.org/abs/2211.03013`.