# The Surprising Effectiveness of Deleting Weights in LLM Post-Training

Jack Lu[*1], Zhenbang Yang[*1], Mike Lasby[2], Tejas Pote[2], Yani Ioannou[2], Mengye Ren[1]
[1]New York University, [2]University of Calgary
https://agenticlearning.ai/deleting-weights/

**Abstract**

How little of a pretrained large language model (LLM) has to change for it to acquire new capabilities? We find that zeroing fewer than 0.05% of its weights, with no other modification, matches full fine-tuning (FFT) at scale across three LLM post-training regimes: reinforcement learning with verifiable rewards, supervised fine-tuning, and on-policy distillation. Our method, *Deletion Tuning*, learns a binary mask over the gate projection of transformer blocks. Two practical advantages follow: the trained mask is hundreds to thousands of times smaller than the corresponding FFT checkpoint, and Deletion Tuning forgets far less, which addresses a central challenge in continual learning. Under reinforcement learning, its prior-task accuracy stays near the base level while FFT, LoRA, and random sparse fine-tuning lose up to 5.4 points. To understand why masking alone suffices, we analyze the geometry of the learned weight changes under reinforcement learning using stable rank, spectrum drift, and principal-weight overlap. Compared with LoRA and random sparse fine-tuning, Deletion Tuning is closest to FFT across the three measures.

## 1 Introduction

To acquire new capabilities during LLM post-training, full fine-tuning can alter every parameter, LoRA (Hu et al., 2022) learns a low-rank update, and the Multiple Ticket Hypothesis (MTH) (Adewuyi et al., 2026) and SHiRA (Bhardwaj et al., 2024) optimize values at selected weight positions. Weight deletion is typically used differently: pruning removes weights while preserving an already-trained model's capabilities for efficient inference (Sun et al., 2024; Frantar and Alistarh, 2023), and the Lottery Ticket Hypothesis finds sparse subnetworks that can be trained to match dense networks (Frankle and Carbin, 2019). Unlike full fine-tuning, LoRA, and sparse updates, deletion writes no new values and only chooses which pretrained weights to keep. Recent work shows that fine-tuned solutions lie close to the pretrained model. RL fine-tuning changes only a small subnetwork (Mukherjee et al., 2025), and task experts lie densely around pretrained weights (Gan and Isola, 2026). We push this observation to its limit across reinforcement learning with verifiable rewards (RLVR), supervised fine-tuning (SFT), and on-policy distillation (OPD), and ask:

*Can a pretrained LLM acquire new capabilities by learning only which weights to set to zero?*

The answer is *yes* across all three post-training regimes. We propose *Deletion Tuning*, which learns only a binary mask over the gate-projection weights of transformer blocks. At deployment, each target weight

[*] Equal contribution.

either retains its pretrained value or is set to zero, so the learned binary mask is the entire adapter. We test Deletion Tuning with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for RLVR, standard SFT, and Self-Distillation Fine-Tuning (SDFT) (Shenfeld et al., 2026a) for OPD. Our experiments span two model families, eight backbones from 0.5B to 30B, and both in-distribution and out-of-distribution evaluation. At the largest model scale tested in each setting, Deletion Tuning matches full fine-tuning (FFT) across reasoning, domain knowledge, and tool-use tasks.

The binary-mask form brings two practical advantages: (i) *the adapter is compact*, with the trained mask hundreds to thousands of times smaller than the corresponding FFT checkpoint and smaller than even LoRA $r$=1 in our main GRPO comparison; and (ii) *Deletion Tuning forgets less*, with its forward KL remaining low and its prior-task accuracy staying near the base level across all three Llama scales. Under FFT, LoRA, and random sparse fine-tuning, forward KL grows and prior-task accuracy falls (Figure 2), consistent with forward KL as a predictor of forgetting (Shenfeld et al., 2026b). Learning new tasks while keeping old ones makes Deletion Tuning a promising candidate for continual learning.

Finally, we examine why Deletion Tuning can match FFT with deletion alone. We confirm that the gains come from learning the mask rather than the Wanda prior used in its initialization. We also find that the geometry of Deletion Tuning's learned weight updates tracks FFT's more closely than LoRA or random sparse fine-tuning across stable rank, spectrum drift, and principal-weight overlap.

Our contributions are as follows:

- **Deletion alone matches FFT.** Deletion Tuning learns which gate-projection weights to delete (Section 4). It is on par with FFT at the largest scale we test in GRPO, SFT, and SDFT, up to a 30B mixture-of-experts model (Sections 5.2 and 5.6). Under GRPO, it also solves problems that the base model fails in all 128 sampled attempts (Section 5.2).
- **Deletion Tuning is compact and forgets less.** Its masks are orders of magnitude smaller than FFT checkpoints in all three regimes (Sections 5.2 and 5.6), and it keeps prior-task accuracy throughout training (Section 5.4).
- **Deletion Tuning's learned updates have FFT-like geometry.** Its weight updates resemble FFT's on three geometric measures (Section 5.5), which helps explain its accuracy.

## 2 Related Work

**LLM post-training.** We study three post-training regimes that differ in where the training signal comes from. Supervised fine-tuning (SFT) imitates fixed labeled targets (Wei et al., 2022; Ouyang et al., 2022; Taori et al., 2023). Reinforcement learning optimizes a reward on the model's own samples, from PPO-based RLHF (Schulman et al., 2017; Ouyang et al., 2022) to GRPO (Shao et al., 2024) with verifiable rewards, the recipe behind DeepSeek-R1 (Guo et al., 2025a). With verifiable rewards, rule-based checkers or LLM verifiers score each sample (Lu et al., 2026a). DPO (Rafailov et al., 2023) instead turns preferences into a closed-form loss. On-policy distillation trains a student on its own rollouts to match a teacher's per-token distribution (Agarwal et al., 2024; Shenfeld et al., 2026a; Zhao et al., 2026), and SDFT uses it to reduce catastrophic forgetting (Kirkpatrick et al., 2017). Self-distillation can also teach reasoning behaviors (Sprague et al., 2026). We use all three to test whether deletion works as a general post-training mechanism.

**Sparsity and selective parameter updates.** One-shot LLM pruning methods like Wanda (Sun et al., 2024) and SparseGPT (Frantar and Alistarh, 2023) extend classical magnitude-based pruning (Han et al., 2015; Zhu and Gupta, 2018) to LLM scale, while continuous relaxations (Louizos et al., 2018; Maddison et al., 2017; Jang et al., 2017) make binary masks differentiable. Unstructured pruning can even retain

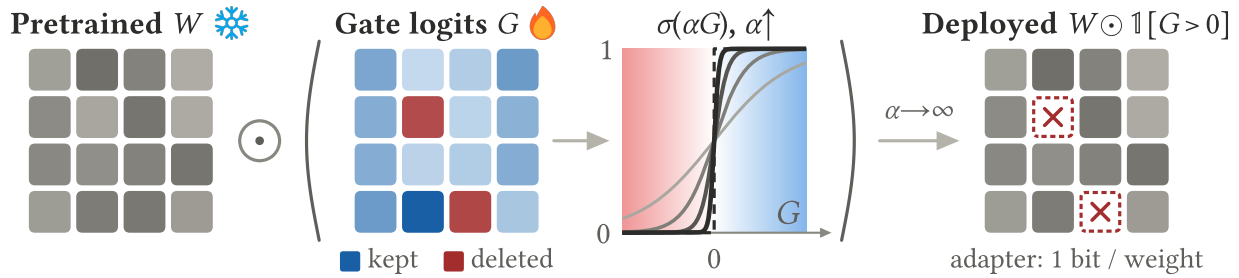



**Figure 1:** ***Deletion Tuning learns which pretrained weights to delete and writes no new values.*** *The pretrained weights $W$ stay frozen, and each is paired with a trainable logit in $G$, the only parameters that post-training updates. During training, each weight is multiplied by $\sigma(\alpha G)$, a smooth mask between 0 and 1. The sharpness $\alpha$ rises over training until the sigmoid becomes a hard threshold at $G = 0$. At deployment, weights with positive logits (blue) keep their pretrained values and weights with negative logits (red) are set to zero (crossed out), so the trained adapter is one bit per weight.*

the test-time scaling of reasoning models (Monjur et al., 2026). Earlier mask learning adapted vision backbones and encoder-only language models to downstream tasks such as image and sentence classification (Mallya et al., 2018; Zhao et al., 2020; Radiya-Dixit and Wang, 2020). Recent mask methods target semi-structured LLM inference sparsity (Fang et al., 2024), refinement of a fine-tuned LLM (Zhang et al., 2025), or vision-language backbones (Zheng et al., 2023). Deletion Tuning instead makes the mask the entire post-training update of a pretrained LLM across RLVR, SFT, and OPD, and it learns how many weights to delete. Sparse weight tuning writes float deltas at selected positions (Guo et al., 2021; Sung et al., 2021; Adewuyi et al., 2026; Liu et al., 2025; Bhardwaj et al., 2024), additive adapters introduce trainable float parameters (Hu et al., 2022; Morris et al., 2026; Sun et al., 2025; Houlsby et al., 2019; Lester et al., 2021; Lu et al., 2026b), and robust fine-tuning methods restrict or regularize movement of pretrained weights under distribution shift (Lee et al., 2023; Tian et al., 2024). Deletion Tuning instead keeps every retained weight at its pretrained value and learns only which gate-projection weights to delete.

**Understanding LLM fine-tuning.** The classical Lottery Ticket Hypothesis (Frankle and Carbin, 2019) posits that randomly initialized networks contain sparse subnetworks that train to comparable accuracy in isolation. Recent work characterizes pretrained-model fine-tuning along three axes. *Density:* task-expert solutions densely populate the neighborhood of pretrained weights (Gan and Isola, 2026). *Geometry:* Mukherjee et al. (2025) report that RL fine-tuning naturally updates only 5 to 30% of parameters across PPO, GRPO, DPO, and several model families. Zhu et al. (2026) show these updates land off the principal weights of $W_0$ while leaving the spectrum largely intact, where principal weights are the top-magnitude entries of the rank-$k$ SVD reconstruction of $W_0$ (Liu et al., 2025). *Forgetting:* Shenfeld et al. (2026b) establish that the forward KL between fine-tuned and base policies on the new task accurately predicts forgetting. Deletion Tuning reproduces the off-principal update pattern and low spectrum drift. Deletion Tuning's low forward KL also predicts the strong prior-task retention we measure (Section 5.4).

## 3 Background

Our experiments use three post-training objectives.

**Supervised fine-tuning (SFT).** Given a dataset $\mathcal{D}$ of prompts $x$ and target responses $y$ of length $L = |y|$, SFT minimizes the negative log-likelihood of the target tokens, which are fixed in advance and never

sampled from the model:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; -\,\mathbb{E}_{(x,y)\sim\mathcal{D}} \sum_{j=1}^{L} \log \pi_\theta(y_j \mid x, y_{<j}). \tag{1}$$

**Group Relative Policy Optimization (GRPO).** For RLVR, we use GRPO (Shao et al., 2024). For each prompt $x$, the policy before the update, $\pi_{\theta_{\mathrm{old}}}$, samples a group of $n$ responses $\{y^i\}_{i=1}^n$, and a verifier assigns each one a reward $r^i = r(x, y^i)$. Rather than learning a value function as in PPO (Schulman et al., 2017), GRPO compares each response with the rest of its group, $A^i = (r^i - \mathrm{mean}(r))/\mathrm{std}(r)$, and gives this advantage to every token of $y^i$ in PPO's clipped objective,

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) \;=\; -\,\mathbb{E}_{x,\{y^i\}} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|y^i|} \sum_{j=1}^{|y^i|} \min\Big(\rho_j^i A^i,\; \mathrm{clip}\big(\rho_j^i,\, 1{-}\epsilon,\, 1{+}\epsilon\big)\, A^i\Big), \tag{2}$$

where $\rho_j^i = \pi_\theta(y_j^i \mid x, y_{<j}^i) \,/\, \pi_{\theta_{\mathrm{old}}}(y_j^i \mid x, y_{<j}^i)$ is the importance ratio for token $j$. GRPO optionally adds a KL penalty to the reference model, weighted by $\beta$.

**Self-Distillation Fine-Tuning (SDFT).** For on-policy distillation, we use SDFT (Shenfeld et al., 2026a), in which the model is its own teacher. The teacher's weights $\theta_T$ are an exponential moving average of the student's weights $\theta$,

$$\theta_T \;\leftarrow\; (1-\gamma)\,\theta_T + \gamma\,\theta, \tag{3}$$

with $\gamma < 1$, and the teacher's prompt also includes $d$, a reference response to the same question, which the student never sees. For a response $y \sim \pi_\theta(\cdot \mid x)$ sampled by the student, the loss is the forward KL between the teacher's and the student's next-token distributions at each position:

$$\mathcal{L}_{\mathrm{SDFT}}(\theta) \;=\; \mathbb{E}_{x,\, y\sim\pi_\theta(\cdot|x)} \sum_{j=1}^{|y|} \mathrm{KL}\Big(\pi_{\theta_T}(\cdot \mid d, x, y_{<j}) \,\big\|\, \pi_\theta(\cdot \mid x, y_{<j})\Big). \tag{4}$$

## 4 Deletion Tuning

*Deletion Tuning* parameterizes fine-tuning as one trainable scalar logit per pretrained weight at a chosen subset of linear layers, with the base model frozen. During training, a sigmoid of each logit is annealed toward a step function, and at deployment each logit becomes a single bit deciding whether the corresponding weight is kept or deleted (set to zero). In pruning terms, Deletion Tuning performs learned unstructured pruning to fine-tune the model.

**Notation.** Let $W \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ denote a target weight matrix in the pretrained model and $G \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ a trainable logit matrix of the same shape, where the sign of $G_{ij}$ at deployment decides whether $W_{ij}$ is kept (positive) or zeroed (negative). We mask one such $W$ per target layer, each with its own independent $G$. In all our experiments, the target is the gate projection of transformer blocks (a single MLP weight per block). We follow GLU-specific pruning methods by treating the gate/up/down pathway as the main coupled sparsification structure (Ma et al., 2023; Guo et al., 2025b): in the Llama and Qwen2.5 backbones we evaluate, SwiGLU MLPs take the form $\mathrm{down}(\mathrm{SiLU}(\mathrm{gate}(x)) \odot \mathrm{up}(x))$, so the gate projection multiplicatively controls the intermediate channels routed through the down projection.

**Forward pass.** At training step $t \in \{0, 1, \ldots, T\}$ (with $T$ the total step count), the layer applies a sigmoid relaxation of the binary masking decision per weight:

$$\widetilde{W}(t) \;=\; W \odot \sigma\big(\alpha(t)\, G\big), \qquad \sigma(x) = \frac{1}{1+e^{-x}}. \tag{5}$$

We call $\sigma(\alpha(t)\, G)$ the *soft mask* and $\mathbf{1}[G > 0]$ the *hard mask*. The sharpness $\alpha(t) > 0$ sets how steep the sigmoid is around $G = 0$. A smaller $\alpha$ widens the band of logits whose soft mask lies strictly between 0 and 1, and raising $\alpha$ pulls the soft mask onto the hard one by the end. We use a linear ramp from $\alpha_{\text{start}} = 10^3$ to $\alpha_{\text{end}} = 10^4$, which we call *mask sharpening*:

$$\alpha(t) \;=\; \alpha_{\text{start}} \;+\; \tfrac{t}{T}(\alpha_{\text{end}} - \alpha_{\text{start}}). \tag{6}$$

At evaluation, the layer applies the hard mask:

$$\widehat{W} \;=\; W \odot \mathbf{1}\big[G > 0\big]. \tag{7}$$

Gradients are computed through the deterministic soft mask in Eq. (5), while the hard mask is used for evaluation and for generating rollouts in on-policy runs. By the end of training, the two agree: deploying the hard mask instead of the soft one changes accuracy by at most 0.08 points.

**Wanda prior initialization.** We initialize $G$ from the Wanda one-shot pruning score (Sun et al., 2024), an importance signal validated as a strong predictor of which base weights are safe to zero. The Wanda score of a weight is its magnitude times the norm of its input activation, $s_{ij} = |W_{ij}| \cdot \|X_j\|_2$, where $X_j$ collects the activations of input channel $j$ in one forward pass over calibration prompts from the task's training distribution. Specifically, we sample each gate logit from one of two Gaussians depending on whether the weight is above or below the $\eta$-quantile of $s$:

$$G_{ij} \;\sim\; \begin{cases} \mathcal{N}(0.03,\; 0.01^2) & \text{if } s_{ij} \text{ is in the top } 1{-}\eta \text{ fraction (high importance),} \\ \mathcal{N}(-0.03,\; 0.01^2) & \text{otherwise (low importance).} \end{cases} \tag{8}$$

At our default $\eta = 1/2048$, used in every main experiment, this zeros approximately $\eta\,\Phi(3) + (1-\eta)\,\Phi(-3) \approx 0.18\%$ of gate-projection weights under the hard mask at initialization, where $\Phi$ is the standard normal CDF.

**Training objective.** We instantiate the policy $\pi_\theta$ of Section 3 by replacing $W$ with $\widetilde{W}(t)$ and keeping pretrained weights elsewhere, so the gate logits $G$ are the only trainable parameters in the model. The training loss $\mathcal{L}_{\text{task}}(\widetilde{W}(t))$ is the regime's own objective from Eqs. (1) to (4), plus the MaskLLM sparse weight regularizer $-\lambda\,\|\widetilde{W}(t)\|_2^2$, which mildly biases the mask toward retaining higher-magnitude weights. We do not constrain $G$ to any fixed sparsity level, leaving the training process to decide it.

- **SFT.** The loss is $\mathcal{L}_{\text{SFT}}$, with $\lambda = 10^{-5}$.
- **GRPO.** The loss is $\mathcal{L}_{\text{GRPO}}$, with $\lambda = 10^{-5}$. Policy rollouts use the hard mask of Eq. (7), while gradients flow into $G$ through the soft mask of Eq. (5).
- **SDFT.** The loss is $\mathcal{L}_{\text{SDFT}}$, with $\lambda = 0$, since the EMA teacher already provides an anchor. Rollouts use the hard mask as in GRPO, and the EMA teacher of Eq. (3) specializes to a teacher gate $G_T \leftarrow (1-\gamma)\, G_T + \gamma\, G$, so the teacher's frozen base weights $W$ stay the same as the student's.

**Compact storage and ease of deployment.** The trained adapter is the hard mask $\mathbf{1}[G > 0]$ at each masked layer, which takes one bit per masked pretrained weight before compression. We bit-pack the booleans with NumPy and compress the bit stream with zstd (Collet and Kucherawy, 2021). Because fewer than 0.05% of all weights are deleted in every GRPO run of Table 1, the bit stream is highly redundant, and zstd compresses it to 1.0 MB at Llama1B and 8.4 MB at Llama8B. This gives Deletion Tuning a practical deployment advantage in compact storage and cheap adapter switching: changing tasks only loads a compact bitmask and scatters zeros into the base weights, rather than loading a separate full model checkpoint.

**Connection to MaskLLM.** MaskLLM (Fang et al., 2024) learns $2:4$ masks for inference speedup with a Gumbel-Softmax selector over $N:M$ candidate patterns. Deletion Tuning has a different goal, expanding model capability, and a different form, with one deterministic logit per weight and no fixed sparsity pattern (Appendix F relates the two formally). Deletion Tuning also adopts MaskLLM's Wanda prior initialization (Eq. (8)) and its sparse weight regularizer.

# 5 Experiments

## 5.1 Experimental Setup

**Tasks, benchmarks, and models.** We study three post-training regimes on two model families, referring to a family as Llama or Qwen and to a size by its suffix (e.g., Llama1B, Qwen7B). Evaluation uses greedy decoding, and we report the best validation accuracy across saved checkpoints.

- **GRPO, in distribution.** We train and evaluate on GSM8K (Cobbe et al., 2021) with Llama1B, Llama3B, and Llama8B (Llama3.2-1B/3B-Instruct and Llama3-8B-Instruct) (Llama Team, AI @ Meta, 2024), validating every 25 optimizer steps.
- **GRPO, out of distribution.** Following SimpleRL-Zoo (Zeng et al., 2025), we train on its math corpus and report the mean over six held-out benchmarks (GSM8K, MATH500, Minerva Math, OlympiadBench, AIME24, AMC23; per-benchmark results in Appendix C) with Qwen0.5B, Qwen3B, and Qwen7B (Qwen2.5 base) (Qwen Team, 2024). At larger scale, we add Qwen14B (Qwen2.5-14B) and the Qwen3-30B-A3B mixture-of-experts model (30B total and 3B active parameters). We validate every 10 optimizer steps.
- **SFT.** We train Llama1B and Llama3B (Llama3.2-1B/3B-Instruct) separately on each of the 14 MMLU-Pro (Wang et al., 2024) subjects and 23 BBEH (Kazemi et al., 2025) subtasks, validate at the end of training, and average accuracy within each suite.
- **SDFT.** We follow the ToolAlpaca (Tang et al., 2023) tool-use and Science settings of Shenfeld et al. (2026a) with Qwen7B (Qwen2.5-7B-Instruct), reporting exact-match accuracy under each task's evaluator. Science is the Chemistry L-3 subset of SciKnowEval (Feng et al., 2024). We validate every 100 optimizer steps on ToolAlpaca and every 25 on Science.

**Training pipelines.** All runs use AdamW with fp32 master weights, bf16 mixed precision, and gradient checkpointing.

- **GRPO.** We use verl (Sheng et al., 2025) with a constant learning rate. GSM8K uses prompt batch 64, $n$=4 rollouts, response length 4096, 3 epochs, and no KL penalty, while SimpleRL-Zoo uses prompt batch 256, $n$=8 rollouts, response length 4096, 150 steps, and KL penalty $10^{-4}$.
- **SFT.** We minimize cross-entropy with a constant learning rate for $T$=400 steps per task at effective batch 4, with a 75/25 train/eval split (per-task counts in Appendix D).
- **SDFT.** We follow Shenfeld et al. (2026a): cosine schedule with 0.1 warmup ratio, prompt batch 32, 2 epochs, EMA mixup $\gamma = 0.01$, and one demonstration prepended to the teacher's input.

**Baselines.** All three baselines use the same data and number of steps as Deletion Tuning.

- **FFT.** Full fine-tuning trains all weights and is the reference we aim to match. Its learning rate is $10^{-6}$ on GSM8K, $5{\times}10^{-7}$ on SimpleRL-Zoo, $10^{-6}$ to $3{\times}10^{-6}$ on SFT (Appendix H), and $5{\times}10^{-5}$ on SDFT as in Shenfeld et al. (2026a).

**Table 1:** ***Deletion Tuning has the smallest adapter at every scale and matches FFT at the largest backbone of each family.*** *Best-checkpoint validation accuracy (%) and adapter size (MB) under GRPO, on GSM8K for Llama and on the six-benchmark SimpleRL-Zoo mean for Qwen. The baselines are full fine-tuning (FFT), low-rank adapters (LoRA), and random sparse fine-tuning (MTH).*

| | Llama, GSM8K | | | | | | Qwen, SimpleRL-Zoo | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1B | | 3B | | 8B | | 0.5B | | 3B | | 7B | |
| **Method** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** |
| Base | 1.21 | 0 | 53.53 | 0 | 60.27 | 0 | 10.44 | 0 | 37.73 | 0 | 40.18 | 0 |
| FFT | 60.50 | 4714 | **82.49** | 12256 | 83.24 | 30633 | 19.01 | 1885 | **44.35** | 11772 | 51.89 | 29051 |
| LoRA r=1 | 59.29 | 2.7 | 82.39 | 5.8 | 83.38 | 10.0 | 20.30 | 2.1 | 43.55 | 7.1 | 52.13 | 9.6 |
| LoRA r=4 | 60.58 | 10.8 | 82.44 | 23.2 | 82.87 | 40.0 | 20.30 | 8.4 | 43.59 | 28.6 | 51.70 | 38.5 |
| MTH 99% | **60.73** | 68.2 | 80.53 | 177.4 | 83.31 | 443.6 | **21.84** | 27.3 | 44.16 | 170.4 | 52.03 | 420.7 |
| MTH 99.95% | 58.61 | 4.1 | 82.17 | 10.7 | 83.07 | 26.6 | 20.16 | 1.7 | 43.87 | 13.7 | 51.24 | 25.3 |
| Deletion Tuning | 54.44 | **1.0** | 81.49 | **3.1** | **83.55** | **8.4** | 17.64 | **0.5** | 41.98 | **2.7** | **52.32** | **6.9** |

- **LoRA.** We train rank-1 and rank-4 adapters (Hu et al., 2022) on all seven projections in each block (q/k/v/o and gate/up/down) with learning rate $10^{-4}$.
- **MTH.** The closest sparse-update method (Adewuyi et al., 2026) trains a random 1% or 0.05% of all parameters (99% and 99.95% sparsity), with the learning rates recommended by Adewuyi et al. (2026) ($10^{-4}$ and $10^{-3}$).

**Deletion Tuning implementation.** We compute the Wanda scores once per task, on 2048 training prompts for GRPO and SFT and on 32 for SDFT.

- **GRPO.** Every main run uses $\eta = 1/2048$, learning rate $3{\times}10^{-4}$, $(\alpha_{\text{start}}, \alpha_{\text{end}}) = (10^3, 10^4)$, and $\lambda = 10^{-5}$, with no tuning per benchmark or model size (Section 5.7 varies $\eta$, learning rate, and $\lambda$).
- **SFT.** We keep the same $\eta$ and $\lambda$, with learning rate $5{\times}10^{-4}$ and $(\alpha_{\text{start}}, \alpha_{\text{end}}) = (10^2, 10^3)$ for every subject and subtask.
- **SDFT.** We set $\eta = 1/2048$ and $\lambda = 0$ and use a hard teacher mask, with learning rate $5{\times}10^{-3}$ and $(10^2, 10^4)$ on ToolAlpaca and $3{\times}10^{-3}$ and $(10^2, 10^3)$ on Science.

**Hardware.** We train on H200 GPUs, using one GPU for backbones up to 3B and multiple GPUs with FSDP (Zhao et al., 2023) for larger models. SFT and the post-hoc analyses use L40S GPUs.

## 5.2 Results on GRPO Post-Training

We train Deletion Tuning and three baselines (FFT, LoRA at ranks 1 and 4, MTH at 99% and 99.95%) on the two GRPO settings of Section 5.1. Table 1 reports best-checkpoint validation accuracy and bytes-on-disk adapter size for six backbones from 0.5B to 8B across two model families. Every Deletion Tuning run deletes fewer than 0.05% of all weights (Appendix B). Appendix A defines adapter size per method.

**Deletion Tuning matches FFT at 7B and 8B.** At the largest backbone in each family, Deletion Tuning scores 83.55% against FFT's 83.24% on Llama8B GSM8K and 52.32% against 51.89% on Qwen7B SimpleRL-Zoo. Tuning FFT's learning rate and KL coefficient does not change this (Appendix H). At these two

**Table 2:** ***Deletion Tuning matches FFT on dense and mixture-of-experts models from 7B to 30B with an adapter thousands of times smaller.*** *Best-checkpoint validation accuracy (%) and adapter size (MB) under GRPO. Qwen14B and Qwen3-30B-A3B use the SimpleRL-Zoo setup of Qwen7B.*

| | **Llama8B** GSM8K | | **Qwen7B** SimpleRL-Zoo | | **Qwen14B** SimpleRL-Zoo | | **Qwen3-30B-A3B** SimpleRL-Zoo | |
|---|---|---|---|---|---|---|---|---|
| **Method** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** |
| Base | 60.27 | 0 | 40.18 | 0 | 44.37 | 0 | 49.41 | 0 |
| FFT | 83.24 | 30633 | 51.89 | 29051 | 54.87 | 56343 | 58.43 | 116471 |
| Deletion Tuning | **83.55** | **8.4** | **52.32** | **6.9** | **55.43** | **11.4** | **58.54** | **29.6** |

**Table 3:** ***Deletion Tuning acquires new capabilities.*** *Pass@k (%), the share of problems solved within k attempts, estimated from* 128 *samples per problem at temperature* 1.0*. A lead at large k means solving problems the base model misses even with many attempts.*

| | **Llama8B, GSM8K** | | | | | **Qwen7B, SimpleRL-Zoo** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | $k$=1 | 2 | 8 | 32 | 128 | $k$=1 | 2 | 8 | 32 | 128 |
| Base | 53.20 | 66.75 | 83.64 | 92.40 | 95.29 | 27.35 | 38.75 | 56.18 | 66.26 | 74.01 |
| Deletion Tuning | **81.75** | **88.51** | **95.75** | **98.44** | **99.62** | **49.39** | **53.22** | **64.64** | **73.23** | **80.08** |

backbones, Deletion Tuning is also competitive with both LoRA ranks and both MTH sparsity levels (Table 1). Since Deletion Tuning can only delete pretrained weights, this restriction may act as a mild regularizer, which could explain why Deletion Tuning matches FFT here. The same restriction pays off more clearly for retention, where Deletion Tuning keeps the prior-task accuracy that FFT loses (Section 5.4).

**The gap to FFT closes with scale.** At smaller backbones, Deletion Tuning recovers 90% to 99% of FFT's accuracy. At larger scale, Deletion Tuning matches FFT on Qwen14B and on the Qwen3-30B-A3B mixture-of-experts model (Table 2). For the mixture-of-experts model, we mask the gate projection inside each of its 128 experts in every layer and leave the router unchanged. These results suggest that deletion becomes more effective as models grow.

**Deletion Tuning acquires new capabilities.** Prior sparsity methods aim to preserve base performance under aggressive pruning (Fang et al., 2024; Sun et al., 2024; Frantar and Alistarh, 2023). Deletion Tuning instead uses sparsity as the learning mechanism itself. On pass@k (Table 3), Deletion Tuning leads the base model at every $k$. Even at $k$=128, the base model's pass@k stays 4.3 and 6.1 points below Deletion Tuning's on Llama8B and Qwen7B.

**Smallest adapter at every scale.** Deletion Tuning's adapter is 0.5 MB at Qwen0.5B and 8.4 MB at Llama8B, 3,600 to 4,700× smaller than the FFT checkpoints and the smallest of any adapter family at every scale. The closest baselines by size are LoRA $r$=1 and MTH at 99.95% sparsity. They are still often several times larger than Deletion Tuning, and MTH at 99% is more than 50× larger in every run. After compression, the mask takes about 0.03 bits per target weight, while every other method stores floating-point values for the weights it changes.

**Table 4:** ***Wanda alone stays near the base model.*** *Validation accuracy (%) of the base model, of Wanda one-shot pruning at the best of eight sparsity levels, and of the best checkpoint of the Deletion Tuning mask trained with GRPO.*

| | Llama, GSM8K | | | Qwen, SimpleRL-Zoo | | |
|---|---|---|---|---|---|---|
| **Method** | 1B | 3B | 8B | 0.5B | 3B | 7B |
| Base | 1.21 | 53.53 | 60.27 | 10.44 | 37.73 | 40.18 |
| Wanda (best $\eta$) | 1.59 | 52.46 | 60.65 | 11.42 | 38.70 | 42.49 |
| Deletion Tuning | **54.44** | **81.49** | **83.55** | **17.64** | **41.98** | **52.32** |

## 5.3 Learning the Mask Drives the Gains

Do Deletion Tuning's gains over the base model come from learning the mask itself, beyond the Wanda prior that initializes the gates? We test this directly. For each run in Table 1, we apply Wanda one-shot pruning without GRPO training at 8 sparsity levels from 1/2 to 1/8192. We evaluate each level on the same validation set and record the *best* accuracy (Appendix E reports every level). Table 4 compares this best-Wanda score with the base model and the trained Deletion Tuning mask.

The best Wanda mask stays within 2.4 points of the base on all six backbones, while Deletion Tuning, starting from the same Wanda scores, gains up to 53.2 points.

## 5.4 Deletion Tuning Forgets Less

Forward KL on the new task predicts how much a fine-tuned model forgets (Shenfeld et al., 2026b). We hypothesize that deletion keeps the perturbation to the base policy small, because each edit can only set a weight to zero and training chooses which few weights to edit, yielding low forward KL and therefore little forgetting. Following Shenfeld et al. (2026b)'s evaluation, Figure 2 tracks forward KL and prior-task accuracy over training for FFT, LoRA, MTH, and Deletion Tuning at the three Llama scales. The prior-task suite has six benchmarks: HellaSwag, MMLU, TruthfulQA, WinoGrande, IFEval, and HumanEval.

By step 348, FFT's forward KL reaches 1.0 on Llama1B and 1.5 on Llama8B, and its prior-task accuracy ends 1.3 to 3.1 points below base. Deletion Tuning's forward KL stays at or below 0.05 at all three scales, and its prior-task accuracy stays within one point of base throughout training. LoRA and MTH train only a small fraction of parameters, yet their forward KL reaches 0.4 to 1.3 and their prior-task accuracy falls up to 5.4 points below base. MTH at 99.95% sparsity trains 0.05% of all parameters, comparable to the 0.035% to 0.046% that Deletion Tuning deletes at these scales (Table 7), suggesting that the small number of edited weights does not explain Deletion Tuning's retention, and that deletion itself keeps forgetting low.

## 5.5 Why Deletion Tuning Works

How can binary masking alone (Section 5.2) match FFT under GRPO at scale? We compare the weight updates of Deletion Tuning and the three baselines at a saved checkpoint on three geometric measures.

**Update geometry.** Let $W_0$ denote the pretrained value of a weight matrix, $W_+$ its effective deployed value at the saved checkpoint (Llama8B step 348, Qwen7B step 100), and $\Delta W = W_+ - W_0$ the weight delta. For FFT, $W_+$ is the trained matrix, and for the adapters it is the merged or masked matrix that runs at inference. We report three quantities, each averaged over all attention and MLP linear layers:

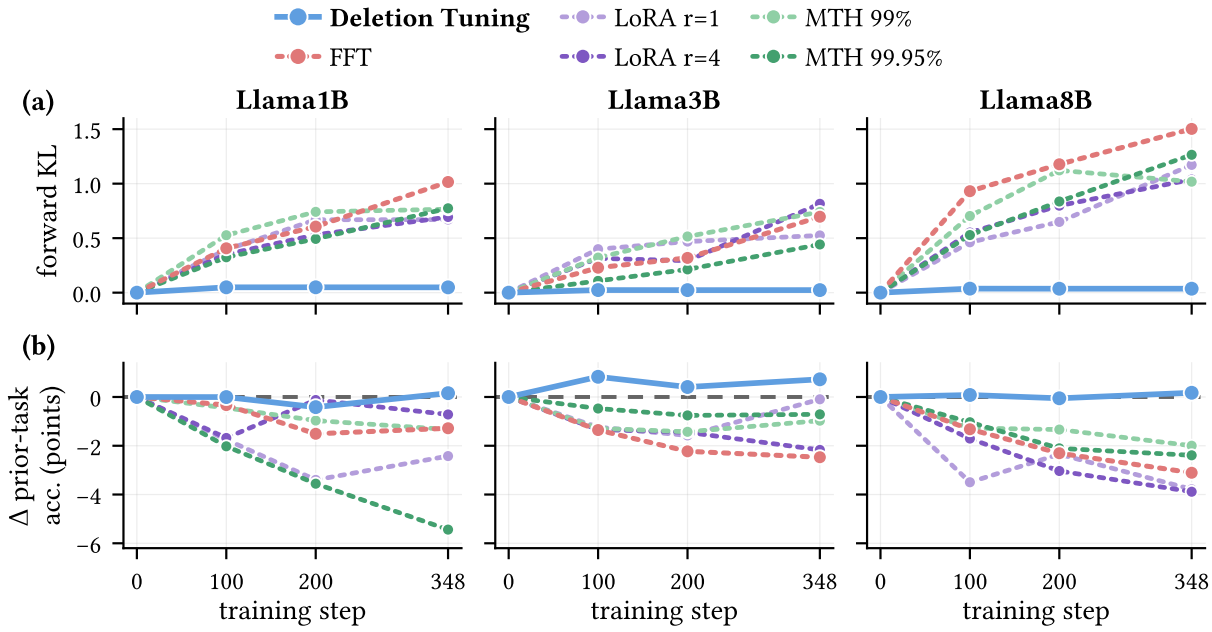



**Figure 2:** ***Deletion Tuning keeps prior-task accuracy at every Llama scale while the baselines drift.*** *(a) Forward KL between the trained and base policies on GSM8K during GRPO training, the forgetting predictor of Shenfeld et al. (2026b). (b) Change in mean accuracy on a six-task prior-task suite from the base model (dashed line).*

- **Stable rank** of $\Delta W$ is $\|\Delta W\|_F^2/\|\Delta W\|_2^2 = \sum_{m=1}^{q} \sigma_m^2/\sigma_1^2$, where $q = \min(d_{\text{out}}, d_{\text{in}})$ and $\sigma_1 \geq \cdots \geq \sigma_q$ are the singular values of $\Delta W$. It counts how many directions the update uses, independent of its scale: it is close to 1 when one direction dominates and close to $k$ when $k$ directions carry similar energy.
- **Spectrum drift** is the normalized singular-value shift $\|\sigma(W_+) - \sigma(W_0)\|_2/\|\sigma(W_0)\|_2$ (Zhu et al., 2026), where $\sigma(W)$ lists the singular values of $W$. It is small when training leaves the singular values close to where they started.
- **Principal-weight overlap** measures the fraction of changed weight positions that fall on what Zhu et al. (2026) call the base model's *principal weights*. The *principal-weight mask*

$$M_{\text{princ}} = \text{Top}_{50\%}\big(|W_0^{(128)}|\big)$$

marks the top 50% entries by magnitude of $W_0^{(128)}$, the best rank-128 approximation of $W_0$. The *update mask* $M_{\text{update}}$ marks positions where the deployed matrix differs from the base by

$$|W_+(i,j) - W_0(i,j)| > 10^{-3} \cdot \max(|W_0(i,j)|, |W_+(i,j)|),$$

the relative tolerance of Zhu et al. (2026), which ignores changes too small to register in bf16. The overlap is $|M_{\text{princ}} \cap M_{\text{update}}|/|M_{\text{update}}|$. Update positions spread uniformly at random would score 0.5, the density of $M_{\text{princ}}$.

Figure 3 shows the three metrics for Deletion Tuning and every baseline setting, and Deletion Tuning is the closest to FFT on the three together. Its stable rank of 66 to 93 sits far below MTH's 410 to 650 and

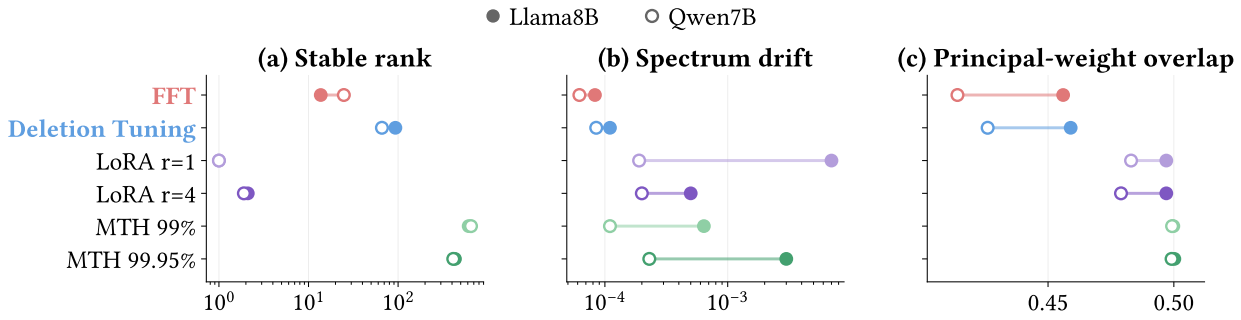



**Figure 3:** ***Deletion Tuning (blue) tracks FFT (red) most closely across three update-geometry metrics.*** *Values at the saved GRPO checkpoint for Llama8B (filled) and Qwen7B (open): (a) stable rank of $\Delta W$, (b) spectrum drift, and (c) principal-weight overlap. Overlap below the random baseline of* 0.5 *is the RLVR signature reported by Zhu et al. (2026).*

above LoRA's 1 to 2. Its spectrum drift stays within 1.4× of FFT's, while LoRA $r$=1 drifts up to 84× more than FFT on Llama8B. Its principal-weight overlap of 0.43 to 0.46 matches FFT's sub-random 0.41 to 0.46. LoRA stays near the random baseline at 0.48 to 0.50, and MTH sits at 0.50.

**Interpretation.** Low spectrum drift and sub-random principal-weight overlap are the RLVR signature reported by Zhu et al. (2026): RLVR leaves the singular values of each weight matrix nearly unchanged and steers its updates away from the principal weights. LoRA caps the rank of its update at $r$, and MTH trains weights chosen at random before training. Deletion Tuning has neither restriction, since it learns which weights to change and has no rank cap. These two freedoms help explain how Deletion Tuning reaches FFT-like geometry even though it never sets a weight to a new non-zero value.

## 5.6 Generalizing Beyond GRPO: Results on SFT and SDFT

Does deletion also work when the training signal is denser or comes from outside the model? GRPO samples from the model itself and rewards whole responses, and RL updates of this kind are known to be sparse (Mukherjee et al., 2025). SFT instead imitates fixed external targets token by token and produces dense updates (Mukherjee et al., 2025), which makes it a harder test for an adapter that can only delete. SDFT sits in between: the student samples its own responses, and a teacher supplies a target distribution at every token. We test Deletion Tuning with SFT on knowledge and reasoning benchmarks (MMLU-Pro and BBEH) and with SDFT on tool use (ToolAlpaca) and science questions (Science), using the setups of Section 5.1.

Under SFT, Deletion Tuning matches FFT at 3B on both MMLU-Pro and BBEH (Table 5). Under SDFT, Deletion Tuning matches FFT on both Qwen7B tasks (Table 6). In both regimes, its adapters are 600 to 4,700× smaller than FFT checkpoints.

## 5.7 Deletion Tuning Is Robust to Hyperparameter Changes

Deletion Tuning works across a range of values for its hyperparameters: initial sparsity $\eta$, learning rate $lr$, and sparse-weight regularization strength $\lambda$. We sweep $\eta \in \{1/2048, 1/4096, 1/8192\}$, $lr \in \{10^{-4}, 3{\times}10^{-4}\}$, and $\lambda \in \{0, 10^{-5}\}$ on Llama8B GSM8K and Qwen7B SimpleRL-Zoo (Tables 13 and 14 in Appendix G). Every setting lands within 1.6 points of the best run on Llama8B and within 3.2 points on Qwen7B, and the single configuration we use for all main runs is at or near the best setting at both backbones. Varying

**Table 5:** ***Under SFT, Deletion Tuning matches FFT at 3B.*** *Best-checkpoint evaluation accuracy (%), averaged over the* 14 *MMLU-Pro subjects and the* 23 *BBEH subtasks, and adapter size (MB) for Llama1B and Llama3B.*

| | MMLU-Pro | | | | BBEH | | | |
|---|---|---|---|---|---|---|---|---|
| | 1B | | 3B | | 1B | | 3B | |
| **Method** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** | **Acc.** | **Size** |
| Base | 10.37 | 0 | 12.63 | 0 | 2.17 | 0 | 8.87 | 0 |
| FFT | 16.97 | 4714 | 21.19 | 12256 | **8.35** | 4714 | **11.39** | 12256 |
| Deletion Tuning | **18.62** | **1.0** | **22.28** | **3.1** | 5.65 | **1.0** | **11.39** | **3.1** |

**Table 6:** ***Under SDFT, Deletion Tuning matches FFT on Qwen7B.*** *Best-checkpoint exact-match accuracy (%) and adapter size (MB) on ToolAlpaca tool use and Science.*

| | ToolAlpaca | | Science | |
|---|---|---|---|---|
| **Method** | **Acc.** | **Size** | **Acc.** | **Size** |
| Base | 41.24 | 0 | 32.35 | 0 |
| FFT | 68.04 | 29051 | **71.20** | 29051 |
| Deletion Tuning | **69.07** | **27.1** | 70.61 | **47.1** |

the initial sparsity $\eta$ by 4× changes accuracy by at most 0.8 points on Llama8B and 1.7 points on Qwen7B at fixed $lr$ and $\lambda$, so the default configuration is a reasonable starting point for new tasks.

## 6 Conclusion

We introduced *Deletion Tuning*, which fine-tunes a model by learning only a binary mask over its gate projections. At the largest scale we test in each regime (Qwen14B and the Qwen3-30B-A3B mixture-of-experts model under GRPO, Qwen7B under SDFT, and Llama3B under SFT), it matches full fine-tuning. The mask is smaller than a rank-1 LoRA and hundreds to thousands of times smaller than an FFT checkpoint. In our GRPO retention runs, Deletion Tuning keeps prior-task accuracy near the base level throughout training, while FFT, LoRA, and MTH lose up to 5.4 points. Its weight updates also resemble FFT's more closely than those of LoRA or MTH across stable rank, spectrum drift, and principal-weight overlap, which helps explain how deletion alone can match FFT. These results suggest that post-training can work by selection alone, keeping nearly every pretrained weight and choosing the few to delete.

**Limitations and Future Work.** Because Deletion Tuning forgets little, a natural next test is continual learning over a sequence of tasks. Training keeps one logit per masked weight, so Deletion Tuning uses less training memory than FFT but more than LoRA. Restricting the logits to a small set of candidate weights could close this gap. The masks are also small and discrete, which aids interpretability: comparing which weights different tasks delete could localize the changes that post-training makes to a model. Masks for different tasks might also be combined, for example by taking their union, to merge skills without further training. Finally, restricting the mask to $N:M$, channel-wise, or block patterns could let one mask both adapt and compress the model.

## Acknowledgments

We thank members of the NYU Agentic Learning AI Lab and the Calgary ML Lab for helpful discussions. JL is supported by an NSERC PGS-D Scholarship. This work was supported in part by Visko Platform, Pionex, a Google TPU Award, a Google ML and Systems Junior Faculty Award, and the Institute of Information & Communications Technology Planning & Evaluation (IITP) with a grant funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research Agreement (No. RS-2024-00469482 & RS-2024-00509279). Computational support was provided by NYU High Performance Computing through its resources, services, and staff expertise. This research was enabled in part by support provided by the Digital Research Alliance of Canada (alliancecan.ca).

ML, TP, and YI gratefully acknowledge the support of Alberta Innovates (ALLRP 577350-22, ALLRP 600038-24), the Natural Sciences and Engineering Research Council of Canada (NSERC) (RGPIN-2022-03120, DGECR-2022-00358), Defence Research and Development Canada (DGDND-2022-03120), and NSERC/Agence Nationale de la Recherche (ANR) (ALLRP 602719-24). This project was undertaken thanks to funding from IVADO and the Canada First Research Excellence Fund. ML additionally acknowledges the support of RBC Borealis through the RBC Borealis AI Global Fellowship Award.

## References

Adewuyi, I., Okibe, S., and Ivanov, V. (2026). The multiple ticket hypothesis: Random sparse subnetworks suffice for RLVR. In *ICML Workshop on High-dimensional Learning Dynamics*.

Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Ramos, S., Geist, M., and Bachem, O. (2024). On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*.

Bhardwaj, K., Pandey, N. P., Priyadarshi, S., Ganapathy, V., Kadambi, S., Esteves, R., Borse, S., Whatmough, P., Garrepalli, R., Baalen, M. V., Teague, H., and Nagel, M. (2024). Sparse high rank adapters. In *NeurIPS*.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Collet, Y. and Kucherawy, M. (2021). Zstandard compression and the `application/zstd` media type. IETF RFC 8878.

Fang, G., Yin, H., Muralidharan, S., Heinrich, G., Pool, J., Kautz, J., Molchanov, P., and Wang, X. (2024). MaskLLM: Learnable semi-structured sparsity for large language models. In *NeurIPS*.

Feng, K., Ding, K., Wang, W., Zhuang, X., Wang, Z., Qin, M., Zhao, Y., Yao, J., Zhang, Q., and Chen, H. (2024). SciKnowEval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098v1*.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.

Frantar, E. and Alistarh, D. (2023). SparseGPT: Massive language models can be accurately pruned in one-shot. In *ICML*.

Gan, Y. and Isola, P. (2026). Neural thickets: Diverse task experts are dense around pretrained weights. In *ICML*.

Guo, D., Rush, A. M., and Kim, Y. (2021). Parameter-efficient transfer learning with diff pruning. In *ACL*.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. (2025a). DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*.

Guo, Z., Kamigaito, H., and Watanabe, T. (2025b). Dependency-aware semi-structured sparsity of GLU variants in large language models. *Transactions on Machine Learning Research*.

Han, S., Pool, J., Tran, J., and Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. In *NeurIPS*.

Houlsby, N., Giurgiu, A., Jastrzębski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In *ICML*.

Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *ICLR*.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with Gumbel-Softmax. In *ICLR*.

Kazemi, M., Fatemi, B., Bansal, H., Palowitch, J., Anastasiou, C., Mehta, S. V., Jain, L. K., Aglietti, V., Jindal, D., Chen, P., Dikkala, N., Tyen, G., Liu, X., Shalit, U., Chiappa, S., Olszewska, K., Tay, Y., Tran, V. Q., Le, Q. V., and Firat, O. (2025). BIG-bench extra hard. In *ACL*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*.

Lee, Y., Chen, A. S., Tajwar, F., Kumar, A., Yao, H., Liang, P., and Finn, C. (2023). Surgical fine-tuning improves adaptation to distribution shifts. In *ICLR*.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

Liu, Z., Pang, T., Balabanov, O., Yang, C., Huang, T., Yin, L., Yang, Y., and Liu, S. (2025). LIFT the veil for the truth: Principal weights emerge after rank reduction for reasoning-focused supervised fine-tuning. In *ICML*.

Llama Team, AI @ Meta (2024). The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through $L_0$ regularization. In *ICLR*.

Lu, J., Teehan, R., Jin, J., and Ren, M. (2026a). When does verification pay off? A closer look at LLMs as solution verifiers. In *ICLR Workshop on AI with Recursive Self-Improvement*.

Lu, J., Teehan, R., Yang, Z., and Ren, M. (2026b). Context tuning for in-context optimization. In *ICML*.

Ma, X., Fang, G., and Wang, X. (2023). LLM-Pruner: On the structural pruning of large language models. In *NeurIPS*.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*.

Mallya, A., Davis, D., and Lazebnik, S. (2018). Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*.

Monjur, O., Nahin, S. K., and Chhabra, A. (2026). Revisiting the effectiveness of LLM pruning for test-time scaling. In *EMNLP Findings*.

Morris, J. X., Mireshghallah, N., Ibrahim, M., and Mahloujifar, S. (2026). Learning to reason in 13 parameters. *arXiv preprint arXiv:2602.04118*.

Mukherjee, S., Yuan, L., Hakkani-Tür, D., and Peng, H. (2025). Reinforcement learning finetunes small subnetworks in large language models. In *NeurIPS*.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback. In *NeurIPS*.

Qwen Team (2024). Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Radiya-Dixit, E. and Wang, X. (2020). How fine can fine-tuning be? Learning efficient language models. In *AISTATS*.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. (2024). DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Shenfeld, I., Damani, M., Hübotter, J., and Agrawal, P. (2026a). Self-distillation enables continual learning. In *ICML*.

Shenfeld, I., Pari, J., and Agrawal, P. (2026b). RL's razor: Why online reinforcement learning forgets less. In *ICLR*.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. (2025). HybridFlow: A flexible and efficient RLHF framework. In *EuroSys*.

Sprague, Z., Lu, J., Wadhwa, M., Keh, S., Ren, M., and Durrett, G. (2026). SkillFactory: Self-distillation for learning cognitive behaviors. In *ICLR*.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. (2024). A simple and effective pruning approach for large language models. In *ICLR*.

Sun, Q., Cetin, E., and Tang, Y. (2025). Transformer-Squared: Self-adaptive LLMs. In *ICLR*.

Sung, Y.-L., Nair, V., and Raffel, C. (2021). Training neural networks with fixed sparse masks. In *NeurIPS*.

Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., Cao, B., and Sun, L. (2023). ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. (2023). Alpaca: A strong, replicable instruction-following model. Stanford CRFM tech report.

Tian, J., Huang, C., and Kira, Z. (2024). Rethinking weight decay for robust fine-tuning of foundation models. In *NeurIPS*.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W. (2024). MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS Datasets and Benchmarks*.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2022). Finetuned language models are zero-shot learners. In *ICLR*.

Zeng, W., Huang, Y., Liu, Q., Liu, W., He, K., Ma, Z., and He, J. (2025). SimpleRL-Zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *COLM*.

Zhang, M., Bai, Y., Wang, H., Wang, Y., Dong, Q., Zhang, Y., and Fu, Y. (2025). Boosting large language models with mask fine-tuning. *arXiv preprint arXiv:2503.22764*.

Zhao, M., Lin, T., Mi, F., Jaggi, M., and Schütze, H. (2020). Masking as an efficient alternative to finetuning for pretrained language models. In *EMNLP*.

Zhao, S., Xie, Z., Liu, M., Huang, J., Pang, G., Chen, F., and Grover, A. (2026). Self-distilled reasoner: On-policy self-distillation for large language models. In *ICML*.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., Desmaison, A., Balioglu, C., Damania, P., Nguyen, B., Chauhan, G., Hao, Y., Mathews, A., and Li, S. (2023). PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*.

Zheng, K., Wu, W., Feng, R., Zhu, K., Liu, J., Zhao, D., Zha, Z.-J., Chen, W., and Shen, Y. (2023). Regularized mask tuning: Uncovering hidden knowledge in pre-trained vision-language models. In *ICCV*.

Zhu, H., Zhang, Z., Huang, H., Su, D., Liu, Z., Zhao, J., Fedorov, I., Pirsiavash, H., Lee, J., Pan, D. Z., Wang, Z., Tian, Y., and Tai, K. S. (2026). The path not taken: RLVR learns off the principals. In *NeurIPS*.

Zhu, M. H. and Gupta, S. (2018). To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *ICLR Workshop*.

# A Adapter Size

The **Size** entries in Tables 1, 2, 5, and 6 (in binary units, 1 GB = 1024 MB) are lower bounds on the bytes on disk needed to rebuild the trained model from the public base checkpoint, computed the same way for every method. We leave out training overhead that deployment discards (file headers, the Adam moments $m$ and $v$, and learning-rate scheduler and RNG state), which would otherwise dominate the smallest adapters.

**Full fine-tuning (FFT).** A fine-tuned LLM checkpoint at fp32:

$$\text{size}(\text{FFT}) = N_{\text{total}} \cdot 4\,\text{bytes},$$

where $N_{\text{total}}$ is the total parameter count. We use fp32 to match the trained master weights, which we store in fp32 for every adapter family.

**LoRA.** The $B$ and $A$ low-rank factor matrices, no bias terms:

$$\text{size}(\text{LoRA-}r) = \sum_{\ell \in \mathcal{T}_{\text{LoRA}}} \left( r \cdot d_{\text{in}}^{(\ell)} + d_{\text{out}}^{(\ell)} \cdot r \right) \cdot 4\,\text{bytes}.$$

**MTH.** The active fp32 values plus a compressed boolean mask of their positions:

$$\text{size}(\text{MTH}) = N_{\text{active}} \cdot 4\,\text{bytes} + \text{zstd}\big(\text{packbits}(\mathbf{1}[\text{active}])\big),$$

where $N_{\text{active}} = (1 - s) \cdot N_{\text{total}}$ for sparsity $s \in \{0.99, 0.9995\}$.

**Deletion Tuning.** A compressed binary mask over the target weights:

$$\text{size}(\text{Deletion Tuning}) = \text{zstd}\big(\text{packbits}(\mathbf{1}[G > 0])\big).$$

No floating-point values are stored: kept weights are read from the public checkpoint, and zeroed weights are zero. At the learned sparsity below 0.05% across the six GRPO runs in Table 1, zstd compresses to roughly 0.03 bits per masked weight.

# B Sparsity Learned by Deletion Tuning per Run

Table 7 reports the fraction of all model weights that Deletion Tuning deletes in the GRPO runs of Table 1 ($\eta = 1/2048$, $\lambda = 10^{-5}$, $lr = 3{\times}10^{-4}$), at the earliest and final saved checkpoints. Every value stays below 0.05%.

**Table 7:** ***Fraction of all weights deleted by Deletion Tuning in the six GRPO runs.*** *Fraction of all model parameters set to zero by the trained mask, at the earliest and final saved checkpoint. Every value is below* 0.05%.

| | **Llama, GSM8K** | | | **Qwen, SimpleRL-Zoo** | | |
|---|---|---|---|---|---|---|
| | **1B** | **3B** | **8B** | **0.5B** | **3B** | **7B** |
| Earliest saved | 0.0342% | 0.0420% | 0.0456% | 0.0393% | 0.0379% | 0.0381% |
| Final saved | 0.0347% | 0.0422% | 0.0459% | 0.0405% | 0.0418% | 0.0415% |

## C Per-Benchmark Qwen2.5 Generalization Results

Table 8 reports the full per-benchmark breakdown that underlies the SimpleRL-Zoo (Qwen) accuracy columns of Table 1.

**Table 8:** ***Qwen2.5 on SimpleRL-Zoo: per-benchmark validation accuracy (%) at the best checkpoint.*** *Six benchmarks: GSM8K, MATH500, Minerva Math, OlympiadBench, AIME24, AMC23.*

| **Method** | GSM8K | MATH500 | Minerva | OlympiadBench | AIME24 | AMC23 |
|---|---|---|---|---|---|---|
| *Qwen0.5B* | | | | | | |
| FFT | 48.07 | 36.00 | 4.78 | 7.73 | 0.00 | 17.50 |
| LoRA r=1 | 49.20 | 33.80 | 7.35 | 8.92 | 0.00 | 22.50 |
| LoRA r=4 | 52.46 | 36.60 | 8.82 | 10.55 | 3.33 | 10.00 |
| MTH 99% | 50.34 | 35.20 | 10.29 | 9.36 | 3.33 | 22.50 |
| MTH 99.95% | 49.73 | 35.20 | 6.25 | 7.28 | 0.00 | 22.50 |
| Deletion Tuning | 44.12 | 33.40 | 5.51 | 6.98 | 3.33 | 12.50 |
| *Qwen3B* | | | | | | |
| FFT | 83.62 | 66.20 | 23.16 | 26.45 | 16.67 | 50.00 |
| LoRA r=1 | 86.81 | 69.80 | 21.69 | 31.35 | 6.67 | 45.00 |
| LoRA r=4 | 85.60 | 67.00 | 22.79 | 28.68 | 10.00 | 47.50 |
| MTH 99% | 86.28 | 67.40 | 22.43 | 29.68 | 6.67 | 52.50 |
| MTH 99.95% | 85.37 | 63.80 | 22.79 | 27.93 | 13.33 | 50.00 |
| Deletion Tuning | 83.78 | 66.20 | 22.79 | 25.78 | 13.33 | 40.00 |
| *Qwen7B* | | | | | | |
| FFT | 91.43 | 78.20 | 33.46 | 39.92 | 13.33 | 55.00 |
| LoRA r=1 | 92.65 | 77.60 | 27.21 | 39.52 | 13.33 | 62.50 |
| LoRA r=4 | 91.66 | 77.00 | 29.04 | 39.97 | 20.00 | 52.50 |
| MTH 99% | 91.21 | 77.20 | 29.78 | 39.82 | 16.67 | 57.50 |
| MTH 99.95% | 92.04 | 75.00 | 31.62 | 38.78 | 20.00 | 50.00 |
| Deletion Tuning | 91.28 | 78.80 | 32.72 | 40.29 | 23.33 | 47.50 |

## D SFT Data Splits

For both MMLU-Pro and BBEH, we split each task's released set 75%/25% into train and held-out evaluation sets. Tables 9 and 10 list per-task counts.

**Table 9:** ***Per-task train/test split for MMLU-Pro*** *under the* 75/25 *split. Total:* 9,018 *train /* 3,014 *test across* 14 *tasks.*

| Category | Train | Test | Category | Train | Test |
|---|---|---|---|---|---|
| biology | 537 | 180 | law | 825 | 276 |
| business | 591 | 198 | math | 1013 | 338 |
| chemistry | 849 | 283 | other | 693 | 231 |
| computer science | 307 | 103 | philosophy | 374 | 125 |
| economics | 633 | 211 | physics | 974 | 325 |
| engineering | 726 | 243 | psychology | 598 | 200 |
| health | 613 | 205 | history | 285 | 96 |

**Table 10:** ***Per-task train/test split for BBEH*** *under the* 75/25 *split. Twenty-two tasks share the* 200*-example pattern. The only outlier is* `disambiguation_qa` *with* 120 *examples. Total:* 3,390 *train /* 1,130 *test across* 23 *tasks.*

| Task group | Train | Test |
|---|---|---|
| 22 tasks (200 examples each) | 150 each | 50 each |
| `disambiguation_qa` (120 examples) | 90 | 30 |

## E Full Wanda Sparsity Sweep

Tables 11 and 12 report the per-sparsity breakdown underlying Section 5.3's best-Wanda picks. $\eta$ is the fraction of gate-projection weights deleted (same convention as Eq. (8)), and the sweep runs from $\eta = 1/2$ (half deleted) to $\eta = 1/8192$ ($\approx 0.012\%$ deleted). Wanda uses noise-free gate init ($G_{ij} = \pm 0.03$, hard-mask sparsity exactly $\eta$), while the trained Deletion Tuning runs of Table 1 use the noisy init of Eq. (8). Qwen values are the six-benchmark mean across GSM8K, MATH500, Minerva Math, OlympiadBench, AIME24, and AMC23. "Base" is the model before fine-tuning. Bold marks the best Wanda accuracy per backbone.

**Table 11:** ***Wanda sweep on Llama (%).***

| $\eta$ | Llama1B | Llama3B | Llama8B |
|---|---|---|---|
| 1/2 | 0.00 | 48.07 | **60.65** |
| 1/8 | 1.36 | 51.71 | 59.14 |
| 1/32 | 1.52 | 51.55 | 58.91 |
| 1/128 | 1.52 | 52.08 | 59.67 |
| 1/512 | 1.44 | 52.16 | 60.05 |
| 1/2048 | **1.59** | 52.16 | 59.74 |
| 1/4096 | **1.59** | 51.63 | 60.05 |
| 1/8192 | 1.36 | **52.46** | 59.67 |
| Base | 1.21 | 53.53 | 60.27 |

**Table 12:** ***Wanda sweep on Qwen (%).***

| $\eta$ | Qwen0.5B | Qwen3B | Qwen7B |
|---|---|---|---|
| 1/2 | 1.92 | 21.06 | 39.39 |
| 1/8 | 10.45 | 36.09 | **42.49** |
| 1/32 | 11.24 | 38.27 | 41.81 |
| 1/128 | 11.21 | 37.71 | 41.90 |
| 1/512 | 10.56 | 37.27 | 40.56 |
| 1/2048 | **11.42** | 38.24 | 40.14 |
| 1/4096 | 11.09 | 38.60 | 40.82 |
| 1/8192 | 10.74 | **38.70** | 41.43 |
| Base | 10.44 | 37.73 | 40.18 |

# F Algebraic Connection to MaskLLM

Deletion Tuning and MaskLLM (Fang et al., 2024) arrive at their forward passes from different starting points: Deletion Tuning from a continuous relaxation of a per-weight binary masking decision (Eq. (5)), and MaskLLM from a Gumbel-Softmax over $N:M$ candidate patterns (Eq. (9)). After three structural modifications to Eq. (9) (binary candidates, no per-group cardinality, no Gumbel noise), the modified MaskLLM forward matches Eq. (5) algebraically.

**MaskLLM's forward.** The base weight matrix is partitioned into groups of $M$ contiguous weights, and each group is handled independently. Within a single group, MaskLLM enumerates $C(M,N) = \binom{M}{N}$ candidate binary patterns $\widehat{M}_c$ with $c \in \{0,\dots,C(M,N)-1\}$, each keeping exactly $N$ of $M$ entries. It places a learnable logit vector $\pi \in \mathbb{R}^{C(M,N)}$ over the candidates and draws a Gumbel-Softmax sample (Jang et al., 2017; Maddison et al., 2017):

$$\tilde{y}_c \;=\; \frac{\exp\big((\kappa\,\pi_c + g_c)/\tau\big)}{\sum_{c'}\exp\big((\kappa\,\pi_{c'} + g_{c'})/\tau\big)}, \qquad g_c \overset{\text{iid}}{\sim} \text{Gumbel}(0,1). \tag{9}$$

The group's soft mask is the candidate-pattern-weighted average $\widetilde{M} = \sum_c \tilde{y}_c \widehat{M}_c$, and the layer forward applies $W \odot \widetilde{M}$ within the group. At evaluation, the hard mask is $\widehat{M}_{c^\star}$ with $c^\star = \arg\max_c \pi_c$. The temperature $\tau$ and logit-scale $\kappa$ both follow linear schedules over training.

**Step 1:** $C(M,N)$=2 **collapses the two-candidate softmax to a sigmoid.** Set $M$=2, $N$=1, so each group has exactly two candidates: $\widehat{M}_0 = (1,0)$ keeps position 0 and $\widehat{M}_1 = (0,1)$ keeps position 1. With only two candidates, the Gumbel-Softmax probability on candidate 0 simplifies to a sigmoid of the logit difference:

$$\tilde{y}_0 = \frac{e^{(\kappa\pi_0+g_0)/\tau}}{e^{(\kappa\pi_0+g_0)/\tau} + e^{(\kappa\pi_1+g_1)/\tau}} = \frac{1}{1+e^{-\frac{1}{\tau}[\kappa(\pi_0-\pi_1)+(g_0-g_1)]}} = \sigma\Big(\tfrac{1}{\tau}\big[\kappa\,(\pi_0-\pi_1)+(g_0-g_1)\big]\Big), \tag{10}$$

where the second equality divides numerator and denominator by the first exponential, and the third uses $\sigma(x) = 1/(1+e^{-x})$. The difference of two independent Gumbel$(0,1)$ variables is Logistic$(0,1)$, so

$$\tilde{y}_0 \;=\; \sigma\big(\tfrac{1}{\tau}[\kappa\,\Delta\pi + \xi]\big), \qquad \Delta\pi := \pi_0 - \pi_1,\;\; \xi \sim \text{Logistic}(0,1). \tag{11}$$

Eq. (11) is the Binary-Concrete distribution (Maddison et al., 2017).

**Step 2: drop the per-group cardinality constraint.** MaskLLM forces each group of $M$ weights to keep exactly $N$ of them. Letting each weight choose its own binary mask value replaces this group categorical with an independent binary masking choice per weight, parameterized by a single logit $G_{ij}$. Each weight then has its own Binary-Concrete (Eq. (11) applied per-weight, with $G_{ij}$ in place of $\Delta\pi$):

$$\widetilde{M}_{ij} \;=\; \sigma\big(\tfrac{1}{\tau}(\kappa\, G_{ij} + \xi_{ij})\big), \qquad \xi_{ij} \sim \text{Logistic}(0,1). \tag{12}$$

**Step 3: drop the noise.** Setting the noise $\xi_{ij}$ to zero in Step 2's per-weight Binary-Concrete gives a deterministic soft mask:

$$\widetilde{M}_{ij} \;=\; \sigma\big(\tfrac{\kappa}{\tau}\, G_{ij}\big). \tag{13}$$

This matches Eq. (5) with the scalar $\alpha(t) := \kappa(t)/\tau(t)$; in our implementation, we schedule this scalar directly and linearly.

## G Deletion Tuning Hyperparameter Sweeps

Tables 13 and 14 give every setting of the sweep in Section 5.7.

**Table 13:** ***Robustness to hyperparameters on Llama8B GSM8K.*** *Best-checkpoint accuracy (%) of Deletion Tuning for each* $(\eta, \lambda, lr)$.

| | $\lambda = 0$ | | $\lambda = 10^{-5}$ | |
|---|---|---|---|---|
| $\eta \setminus lr$ | $10^{-4}$ | $3{\times}10^{-4}$ | $10^{-4}$ | $3{\times}10^{-4}$ |
| 1/2048 | 83.40 | 82.49 | 82.34 | **83.55** |
| 1/4096 | 83.02 | 82.03 | 83.09 | 83.09 |
| 1/8192 | 83.40 | 82.49 | 83.09 | 83.02 |

**Table 14:** ***Robustness to hyperparameters on Qwen7B SimpleRL-Zoo.*** *Best-checkpoint accuracy (%) of Deletion Tuning for each* $(\eta, \lambda, lr)$.

| | $\lambda = 0$ | | $\lambda = 10^{-5}$ | |
|---|---|---|---|---|
| $\eta \setminus lr$ | $10^{-4}$ | $3{\times}10^{-4}$ | $10^{-4}$ | $3{\times}10^{-4}$ |
| 1/2048 | 51.79 | 51.92 | 49.60 | 52.32 |
| 1/4096 | 52.23 | **52.80** | 50.29 | 51.91 |
| 1/8192 | 51.14 | 52.10 | 51.29 | 52.34 |

## H FFT Hyperparameter Sweeps

To make sure the FFT baseline is not under-tuned, we sweep its learning rate under GRPO and SFT and its KL coefficient under GRPO. All other settings follow the main runs. None of these settings changes the comparison with Deletion Tuning (Tables 15 and 16).

**Table 15:** ***FFT sweep under GRPO.*** *Best-checkpoint validation accuracy (%), as in Table 1.*

| Method | Learning rate | KL coefficient | Llama8B, GSM8K | Qwen7B, SimpleRL-Zoo |
|---|---|---|---|---|
| FFT | $5{\times}10^{-7}$ | 0 | 83.23 | 50.55 |
| FFT | $5{\times}10^{-7}$ | $10^{-4}$ | 82.55 | 51.89 |
| FFT | $10^{-6}$ | 0 | 83.24 | 51.93 |
| FFT | $10^{-6}$ | $10^{-4}$ | 81.89 | 50.79 |
| Deletion Tuning | $3{\times}10^{-4}$ | as in Table 1 | **83.55** | **52.32** |

**Table 16:** ***FFT learning-rate sweep under SFT, Llama3B MMLU-Pro.*** *Best-checkpoint accuracy (%), as in Table 5.*

| Method | Learning rate | Accuracy |
|---|---|---|
| Base | – | 12.63 |
| FFT | $10^{-5}$ | 19.35 |
| FFT | $3\times10^{-6}$ | 21.19 |
| FFT | $10^{-6}$ | 18.35 |
| Deletion Tuning | $5\times10^{-4}$ | **22.28** |