The Surprising Effectiveness of Deleting Weights in LLM Reasoning and Adaptation

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

[*]Equal contribution

Abstract

How little of a pretrained large language model has to change for it to generalize to new tasks and acquire new reasoning capabilities? We find that zeroing fewer than 0.05% of its weights, with no other modification, matches full-parameter fine-tuning (FPT) at scale across the three dominant LLM fine-tuning paradigms: on-policy reinforcement learning with verifiable rewards (GRPO), supervised fine-tuning (SFT), and on-policy distillation (SDFT). Our method *Bit-Mask Tuning* (*BMT*), a learnable binary keep/zero mask over the gate projection of transformer blocks, reaches FPT accuracy at our largest backbones with the GRPO gap closing monotonically across model sizes from 0.5B to 8B; the trained mask is several $1{,}000\times$ smaller than the full-parameter checkpoint, and *BMT* retains prior-task accuracy throughout training while FPT measurably drifts. To understand why masking alone suffices, we study the learning dynamics through update-geometry analysis on GRPO and find that *BMT*'s weight delta lands closer to full-parameter updates than that of any other adapter on three measures: effective rank, spectrum drift, and principal-weight overlap. In sum, deletion alone produces a tiny yet high-performing adapter, reduced forgetting, and an update geometry that tracks FPT's more closely than any baseline we evaluate.

1. Introduction

Fine-tuning a pretrained LLM is normally additive. Full-parameter methods (Shao et al., 2024; Zeng et al., 2025) rewrite every weight, LoRA (Hu et al., 2022) adds a low-rank delta on top of a frozen base, and sparse-update methods (Adewuyi et al., 2026; Bhardwaj et al., 2024) inject new values at chosen positions. We ask whether the addition is necessary at all. Prior sparsity work prunes already-trained models for inference efficiency (Sun et al., 2024; Frantar & Alistarh, 2023; Fang et al., 2024), and the lottery-ticket hypothesis (Frankle & Carbin, 2019) posits trainable sparse subnetworks, but to our knowledge, no method has tested whether erasure alone can serve as the entire fine-tuning mechanism across modern post-training regimes (on-policy RL, SFT, and distillation). We push the question to its limit: *can a pretrained LLM acquire new reasoning capabilities if the only change permitted is which weights to zero?*

The answer is *yes*, and a single parameterization works across regimes. We call it *Bit-Mask Tuning* (*BMT*, §4): the only learnable parameters form a binary keep/zero mask over the gate projection of transformer blocks, with every base weight kept bit-identical to pretraining or zeroed. The trained adapter is just the positions of the deletions, a structural, implicit regularization that restricts fine-tuning to subsets of the pretrained weight set.

In extensive experiments spanning two model families, six backbone sizes (0.5B to 8B), and four benchmark suites covering both in-distribution and out-of-distribution evaluation, *BMT* reaches full-parameter fine-tuning (FPT) accuracy at our largest backbone in each of the three regimes. Under GRPO on math, the gap closes monotonically as the backbone scales: zeroing fewer than 0.05% of model parameters matches full-parameter GRPO on Llama8B (Llama3-8B-Instruct, GSM8K) and Qwen7B (Qwen2.5 base, SimpleRL-Zoo averaged over GSM8K, MATH500, Minerva, OlympiadBench, AIME24, AMC23). Under on-policy distillation (SDFT (Shenfeld et al., 2026a)) on Qwen7B (Qwen2.5-7B-Instruct, ToolAlpaca), and under SFT on Llama3B (Llama3.2-3B-Instruct) on MMLU-Pro (Wang et al., 2024) and BBEH (Kazemi et al., 2025), *BMT* again ties the full-parameter baseline (§5.6).

Accepted to FoGen 2026: Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning, an ICML 2026 workshop (non-archival).

Two practical benefits follow from the binary-mask form: (i) *the adapter compresses to a negligible footprint*, since the trained mask carries one bit per masked weight at $< 0.05\%$ learned sparsity, so after entropy coding it is several $1{,}000\times$ smaller than the full-parameter checkpoint and smaller than even LoRA $r{=}1$ (§5.2); and (ii) *BMT forgets substantially less than FPT*: forward KL on the new task plateaus an order of magnitude below FPT (Figure 2), and prior-task accuracy holds throughout training while FPT loses 1–3% on the same retention suite, consistent with the forgetting predictor of Shenfeld et al. (2026b).

Why does masking alone match FPT? Two findings answer this, both diagnosed on the GRPO runs. *Learning drives the gain, not the prior:* Wanda one-shot pruning (Sun et al., 2024) alone moves base accuracy by at most 2.4%, while training the gates from the same Wanda initialization lifts accuracy by up to 53%. *The update geometry coincides with FPT's:* though a binary keep/zero rule might be expected to produce updates unlike full-parameter ones, *BMT*'s weight delta is more FPT-like than that of every other adapter we evaluate on effective rank, spectrum drift, and principal-weight overlap. LoRA and high-sparsity MTH drift away from FPT in distinct ways.

We make the following contributions:

- **Sparse tuning method and empirical phenomenon.** *Bit-Mask Tuning* (*BMT*, §4), a learnable binary keep/zero mask over the gate projection of transformer blocks, matches full-parameter fine-tuning at the largest backbones across GRPO, SFT, and SDFT (Llama8B GSM8K, Qwen7B SimpleRL-Zoo, Qwen7B ToolAlpaca, Llama3B MMLU-Pro and BBEH, §5.2 and 5.6). The gap to FPT closes monotonically across GRPO model sizes, and results are stable across our hyperparameter sweep (§5.7).
- ***BMT* is tiny and forgets less than FPT.** The trained *BMT* mask is several $1{,}000\times$ smaller than the full-parameter checkpoint and smaller than even LoRA $r{=}1$ (§5.2). *BMT* also forgets substantially less than FPT, retaining prior-task accuracy throughout training while FPT loses 1–3% on the same retention suite (§5.4).
- ***BMT*'s update geometry is close to FPT's.** On the GRPO checkpoints, *BMT*'s weight delta is closer to full-parameter updates than that of LoRA $r{=}1$, LoRA $r{=}4$, MTH at 99%, or MTH at 99.95% on every measure we check (effective rank, spectrum drift, and principal-weight overlap), surfacing an explanation for *BMT*'s efficacy (§5.3–5.5).

2. Related work

LLM reasoning and adaptation. Three paradigms dominate LLM fine-tuning. *Supervised fine-tuning (SFT)* (Wei et al., 2022; Ouyang et al., 2022; Taori et al., 2023) optimizes log-likelihood on labeled targets. *Reinforcement learning* spans PPO (Schulman et al., 2017) behind RLHF (Ouyang et al., 2022), DPO (Rafailov et al., 2023) reformulating preferences as a closed-form loss, and GRPO (Shao et al., 2024) driving verifiable-reward gains exemplified by DeepSeek-R1 (Guo et al., 2025) and enabling test-time scaling (Snell et al., 2025; Sprague et al., 2026; Lu et al., 2026a). *On-policy distillation* aligns student rollouts with a teacher's per-token distribution (Agarwal et al., 2024; Shenfeld et al., 2026a; Zhao et al., 2026) while mitigating catastrophic forgetting (Kirkpatrick et al., 2017). We evaluate *Bit-Mask Tuning* across all three paradigms.

Sparsity in neural networks and LLMs. One-shot LLM pruning methods like Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023) extend classical magnitude-based pruning (Han et al., 2015; Zhu & Gupta, 2017) to LLM scale, while continuous-relaxation methods (Louizos et al., 2018; Maddison et al., 2017; Jang et al., 2017) make binary masks differentiable. Three methods learn binary masks on frozen weights: RAMT (Zheng et al., 2023) on vision-language backbones, MaskLLM (Fang et al., 2024) for $N{:}M$ sparsity at LLM scale, and MFT (Zhang et al., 2025) as a post-fine-tuning refinement under next-token prediction. Sparse weight tuning instead writes float deltas at chosen sparse subsets (Guo et al., 2021; Sung et al., 2021; Adewuyi et al., 2026; Liu et al., 2025; Bhardwaj et al., 2024), and additive adapters (Hu et al., 2022; Morris et al., 2026; Sun et al., 2025; Houlsby et al., 2019; Lester et al., 2021; Lu et al., 2026b) introduce trainable float parameters rather than selecting which existing weights to keep. *Bit-Mask Tuning* pushes the learnable-mask branch to RLVR-driven LLM post-training, with selective zeroing as the only weight modification.

Understanding LLM fine-tuning. The classical Lottery Ticket Hypothesis (Frankle & 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:* Neural Thickets (Gan & Isola, 2026) shows that task-expert solutions densely populate the neighborhood of pretrained weights, with density growing as the backbone scales, directly motivating our scaling result. *Geometry:* Mukherjee et al. (2025) report that RL fine-tuning naturally

updates only 5–30% of parameters across PPO, GRPO, DPO, and several model families, and Zhu et al. (2025) 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. *BMT* reproduces the off-principal, spectrum-preserving signature, and our analysis additionally identifies low effective rank in FPT updates (§5.5). *BMT*'s low forward KL also predicts the strong prior-task retention we measure (§5.4).

To our knowledge, *Bit-Mask Tuning* is the first method in which a learnable binary mask alone (§5.2) suffices for LLM fine-tuning across RLVR, SFT, and on-policy distillation.

3. Background

Bit-Mask Tuning is a parameterization, not a training algorithm. We evaluate it across the three regimes most relevant to current LLM post-training: supervised fine-tuning (SFT) on labeled targets, reinforcement learning with verifiable rewards (RLVR) via GRPO (Shao et al., 2024), and on-policy distillation via SDFT (Shenfeld et al., 2026a).

Supervised fine-tuning (SFT). For labeled (x, y) pairs, SFT minimizes per-token cross-entropy of the target conditioned on the prompt and prefix:

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

The targets are drawn from a fixed dataset $\mathcal{D}$, off-policy with respect to the current model π_θ.

Group-relative policy optimization (GRPO). GRPO (Shao et al., 2024) is a critic-free policy-gradient algorithm developed for RLVR on math and general reasoning. For each prompt x, the rollout policy $\pi_{\theta_{\text{old}}}$ generates a group of n on-policy completions $\{y^i\}_{i=1}^n$, each scored by a verifiable reward $r^i = r(x, y^i)$. The group-normalized advantage

$$A^i \;=\; \frac{r^i - \text{mean}(r)}{\text{std}(r)} \tag{2}$$

replaces the value baseline used in PPO (Schulman et al., 2017). The actor objective is the standard PPO clipped policy ratio against A^i at every token of every rollout:

$$\mathcal{L}_{\text{GRPO}}(\theta) \;=\; -\,\mathbb{E}_{x,\,\{y^i\}} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|y^i|}\sum_{j=1}^{|y^i|} \min\Big(\rho_j^i A^i,\; \text{clip}\big(\rho_j^i,\, 1{-}\epsilon,\, 1{+}\epsilon\big)\, A^i\Big), \tag{3}$$

with the per-token importance ratio $\rho_j^i \;=\; \pi_\theta(y_j^i \mid x, y_{<j}^i) \,/\, \pi_{\theta_{\text{old}}}(y_j^i \mid x, y_{<j}^i)$. GRPO optionally adds a per-token KL-to-reference penalty with coefficient β.

On-policy self-distillation fine-tuning (SDFT). SDFT (Shenfeld et al., 2026a) is a continual-learning method in which the same model serves as both student and teacher. The teacher's parameters θ_T track the student's θ by an exponential moving average,

$$\theta_T \;\leftarrow\; (1-\gamma)\,\theta_T + \gamma\,\theta, \tag{4}$$

with $\gamma < 1$ so the teacher tracks the student slowly. For each training prompt x, a small task demonstration d is prepended to the teacher's input but not the student's. The student samples an on-policy continuation $y \sim \pi_\theta(\cdot \mid x)$, and the gradient signal is per-token forward KL[1] from the teacher's distribution conditioned on (d, x) to the student's distribution conditioned on x:

$$\mathcal{L}_{\text{SDFT}}(\theta) \;=\; \mathbb{E}_{x,\,y\sim\pi_\theta(\cdot|x)} \sum_{j=1}^{|y|} \text{KL}\Big(\pi_{\theta_T}(\cdot \mid d, x, y_{<j}) \;\big\|\; \pi_\theta(\cdot \mid x, y_{<j})\Big). \tag{5}$$

The combination is on-policy in the student's response distribution and pulls the student toward the response distribution the teacher produces when shown the demonstration d.

[1]We follow Shenfeld et al. (2026a)'s public implementation, which uses per-token forward KL; the authors note that the arXiv reverse-KL form does not match their training code.

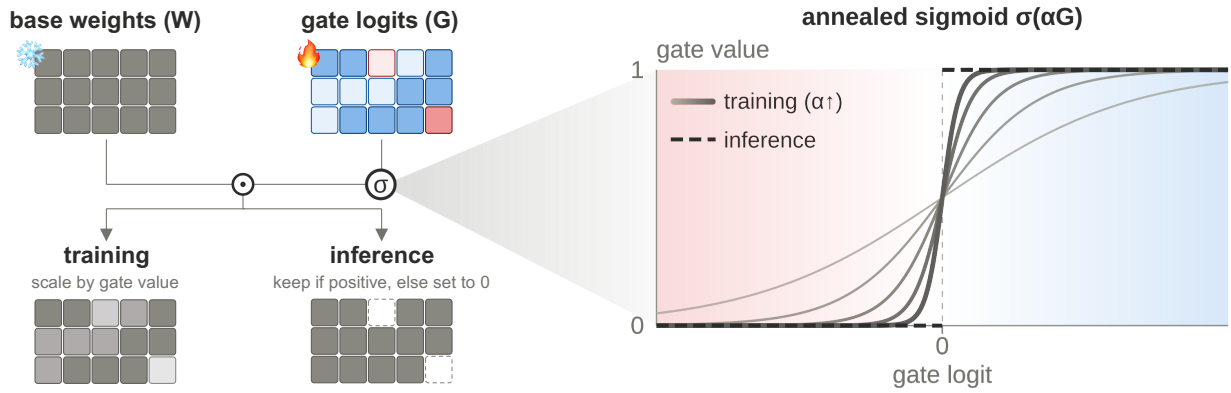

Figure 1. ***Bit-Mask Tuning*** **forward pass.** A trainable gate logit G is combined with the frozen base weight W via an annealed sigmoid. *Left:* during training, each weight in W is scaled by a gate value in $[0, 1]$, while at inference the weight is kept if its gate logit is positive and zeroed otherwise. *Right:* the annealed sigmoid function sharpens over training with increasing α toward the step function used at inference.

4. Bit-Mask Tuning

Bit-Mask Tuning parameterizes fine-tuning as one trainable scalar logit per pretrained weight at a chosen subset of linear layers, with the base model frozen. At deployment, each logit hardens into a single bit deciding whether the corresponding weight is kept or zeroed.

Notation. Let $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denote a target weight matrix in the pretrained model and $G \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ 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).

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 keep/zero decision per weight:

$$\widetilde{W}(t) \;=\; W \odot \sigma\big(\alpha(t)\, G\big), \qquad \sigma(x) = \frac{1}{1 + e^{-x}}. \tag{6}$$

We call $\sigma(\alpha(t)\, G)$ the *soft mask* and $\mathbf{1}[G > 0]$ the *hard mask*. The sharpness $\alpha(t) > 0$ controls how steep the sigmoid is around $G{=}0$: smaller α keeps the soft mask away from saturation so gradients flow across a wide range of G in early and mid training, while larger α collapses the soft mask onto the hard one near the end. We use a linear ramp from $\alpha_{\text{start}} = 10^3$ to $\alpha_{\text{end}} = 10^4$, which we call *mask sharpening*:

$$\alpha(t) \;=\; \alpha_{\text{start}} \;+\; \tfrac{t}{T}(\alpha_{\text{end}} - \alpha_{\text{start}}). \tag{7}$$

At evaluation, the layer applies the hard mask:

$$\widehat{W} \;=\; W \odot \mathbf{1}\big[G > 0\big]. \tag{8}$$

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 score $s_{ij} = |W_{ij}| \cdot \|X_j\|_2$ is computed in a single calibration forward pass over 2048 prompts drawn from the task's training distribution, adding negligible compute overhead. Specifically, we sample each gate logit from one of two Gaussians depending on whether the weight is above or below the η-quantile of s:

$$G_{ij} \;\sim\; \begin{cases} \mathcal{N}(0.03,\ 0.01^2) & \text{if } s_{ij} \text{ is in the top } 1{-}\eta \text{ fraction (high importance)}, \\ \mathcal{N}(-0.03,\ 0.01^2) & \text{otherwise (low importance)}. \end{cases} \tag{9}$$

At our default $\eta = 1/2048$ used throughout the GRPO runs, this zeros approximately $\eta\,\Phi(3) + (1-\eta)\,\Phi(-3) \approx 0.18\%$ of gate-projection weights under the hard mask at initialization.

Training objective. We instantiate the policy π_θ of §3 by replacing W with $\widetilde{W}(t)$ and keeping pretrained weights elsewhere. The gate logits G are the only trainable parameters in the model. The training loss $\mathcal{L}_{\text{task}}(\widetilde{W}(t))$ is one of

$\mathcal{L}_{\text{SFT}}$, $\mathcal{L}_{\text{GRPO}}$, or $\mathcal{L}_{\text{SDFT}}$ from Eqs. (1)–(5). For GRPO and SDFT, policy rollouts use the hard mask of Eq. (8), while gradients flow into G through the soft mask of Eq. (6). For SDFT, the EMA teacher of Eq. (4) 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. We additionally regularize the loss with the MaskLLM sparse weight regularizer $-\lambda \, \|\widetilde{W}(t)\|_2^2$ at $\lambda = 10^{-5}$ for GRPO and SFT, which mildly biases the mask toward retaining higher-magnitude weights. For SDFT we set $\lambda = 0$ since the EMA teacher already provides an anchor. We do not constrain G to any fixed sparsity level, leaving the training process to decide it.

Compact storage and ease of deployment. The trained adapter is the hard mask $\mathbf{1}[G > 0]$ at each masked layer, naively one bit per masked pretrained weight. We bit-pack the booleans with `numpy.packbits` and compress the bit stream with zstd (Collet & Kucherawy, 2018). At the learned total-network sparsity below 0.05% we observe across the six GRPO runs of §5.5, the bit stream is so redundant that zstd compresses it to just 1.0 MB at Llama1B and 8.4 MB at Llama8B (Table 1), several $1{,}000\times$ smaller than the corresponding full-parameter checkpoint. Switching and deploying adapters requires only a scatter operation that zeroes base weights at masked positions, structurally cheaper than serving a full checkpoint or LoRA's matmul merges.

Connection to MaskLLM. MaskLLM (Fang et al., 2024) is a learnable-mask method targeting inference speedup, with a Gumbel-Softmax selector over $N{:}M$ candidate patterns at $2{:}4$ sparsity. Although the motivations differ (*BMT* expands model capability) and the formulations appear distinct, Appendix G shows that MaskLLM's structured forward reduces to *BMT*'s after three structural simplifications. *BMT* also adopts MaskLLM's Wanda prior initialization (Eq. (9)) and its sparse weight regularizer.

5. Experiments

5.1. Experimental setup

Tasks, benchmarks, and models. We run two GRPO experiments. The first is in-distribution training and evaluation on GSM8K (Cobbe et al., 2021) with Llama1B, Llama3B, and Llama8B (Llama3.2-1B/3B-Instruct and Llama3-8B-Instruct) (Grattafiori et al., 2024). The second tests out-of-distribution generalization and cross-benchmark transfer following SimpleRL-Zoo (Zeng et al., 2025): we train on their math corpus and evaluate on six held-out benchmarks (GSM8K, MATH500, Minerva-Math, OlympiadBench, AIME24, AMC23) with Qwen0.5B, Qwen3B, and Qwen7B (Qwen2.5 base) (Yang et al., 2024), reporting the six-benchmark mean (per-benchmark breakdown in Appendix D). SFT uses MMLU-Pro (Wang et al., 2024) with 14 subjects and BBEH (Kazemi et al., 2025) with 23 subtasks on Llama1B and Llama3B (Llama3.2-1B/3B-Instruct), running one training run per subject/subtask and averaging accuracy within each suite. SDFT follows the ToolAlpaca tool-use setting of Shenfeld et al. (2026a) with Qwen7B (Qwen2.5-7B-Instruct) and reports accuracy on the tool-call test split. We refer to a model family generically as Llama or Qwen and to a specific size with the size suffix (e.g. Llama1B, Qwen7B). Evaluation uses greedy sampling. We validate every 25 optimizer steps on GSM8K and every 10 on SimpleRL-Zoo for GRPO, every 100 for SDFT, and at the end of training for SFT, and report best validation accuracy across saved checkpoints throughout.

Training pipelines. All runs use AdamW with fp32 master weights, bf16 mixed precision, gradient checkpointing, and seed 0. GRPO and SFT use a constant learning-rate schedule, and SDFT uses cosine with 0.1 warmup ratio following Shenfeld et al. (2026a). GRPO runs through verl (Sheng et al., 2025): 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 uses cross-entropy for $T{=}400$ per-task steps at effective batch 4, with a $75/25$ train/eval split (per-task counts in Appendix E). SDFT follows Shenfeld et al. (2026a): prompt batch 32, 2 epochs, EMA mixup $\gamma = 0.01$, one demonstration prepended to the teacher's input only, on-policy student rollouts.

Baselines. We compare *BMT* against three baselines: (i) full-parameter fine-tuning (FPT) where every weight is trainable, (ii) LoRA (Hu et al., 2022) at ranks 1 and 4 on all seven projections per block (q/k/v/o and gate/up/down), and (iii) MTH (Adewuyi et al., 2026) at 99% and 99.95% sparsity following Adewuyi et al. (2026), applied to every parameter. *BMT* masks only the gate projection of transformer blocks. For fair comparison, all methods share identical dataset splits, base model, hyperparameters, total optimizer steps, and store trainable parameters in fp32. LoRA uses standard learning rate 10^{-4}, MTH uses the per-sparsity rates of Adewuyi et al. (2026) (10^{-4} at 99%, 10^{-3} at 99.95%), and FPT uses 10^{-6} on GSM8K and SFT, $5{\times}10^{-7}$ on SimpleRL-Zoo, and $5{\times}10^{-5}$ on SDFT (the paper-default FPT rate of Shenfeld et al. (2026a)). We additionally implemented TinyLoRA (Morris et al., 2026) but exclude it from comparisons

Table 1. **Combined performance across the two GRPO settings.** Best-checkpoint validation accuracy (%) and adapter size (MB) for six backbones. *BMT* produces the smallest adapter at every scale and matches FPT at the largest backbone in both settings.

	Llama, GSM8K						Qwen, SimpleRL-Zoo					
	1B		3B		8B		0.5B		3B		7B	
Method	Acc. (%)	Size (MB)	Acc. (%)	Size (MB)	Acc. (%)	Size (MB)	Acc. (%)	Size (MB)	Acc. (%)	Size (MB)	Acc. (%)	Size (MB)
Base	1.21	0	53.53	0	60.27	0	10.44	0	37.73	0	40.18	0
FPT	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
BMT	54.44	**1.0**	81.49	**3.1**	**83.55**	**8.4**	17.64	**0.5**	41.98	**2.7**	**52.32**	**6.9**

because its training is unstable and learning-rate-sensitive, requiring extensive per-model and per-benchmark tuning.

BMT implementation. Across all regimes, G is stored in fp32, and Wanda calibration runs once per task on 2048 prompts drawn from the training distribution. All main GRPO runs share a *single* configuration with initial sparsity $\eta = 1/2048$, learning rate $3{\times}10^{-4}$, sharpness $(\alpha_{\text{start}}, \alpha_{\text{end}}) = (10^3, 10^4)$, and sparse weight regularizer $\lambda = 10^{-5}$. *We do not tune per benchmark or per model scale* across Llama1B/3B/8B on GSM8K or Qwen0.5B/3B/7B on SimpleRL-Zoo, and §5.7 confirms robustness across η, learning rate, and λ. SFT reuses the same η and λ, raises learning rate to $5{\times}10^{-4}$, and lowers sharpness to $(\alpha_{\text{start}}, \alpha_{\text{end}}) = (10^2, 10^3)$, again with a single configuration covering every SFT run. SDFT keeps $\eta = 1/2048$, sets learning rate to $5{\times}10^{-3}$ since forward-KL gradients are small, ramps sharpness from 10^2 to 10^4, uses a hard teacher mask, and sets $\lambda = 0$. We submit the *BMT* implementation and reproduction instructions as supplementary code.

Hardware. We use one H200 for GRPO and SDFT backbones $\leq$ 3B and two with FSDP (Zhao et al., 2023) sharding for larger backbones, with each run taking up to 16 hours and totaling more than 2,000 H200-hours across all configurations. SFT training and all post-hoc analyses run on L40S, with 3B FPT FSDP-sharded across $2\times$ L40S with CPU offload.

5.2. Results on GRPO post-training

We train *BMT* and three baselines (FPT, LoRA at ranks 1 and 4, MTH at 99% and 99.95%) on the two GRPO settings of §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. All *BMT* runs realize total-network sparsity below 0.05%, which grows from initialization through training (evidence that the mask actively adapts its sparsity level, Appendix C). Appendix B defines adapter size per method.

The gap to FPT closes monotonically with model capacity. At the smallest backbones, *BMT* recovers $54.44/60.50 \approx 90\%$ of FPT's accuracy on Llama1B and $17.64/19.01 \approx 93\%$ on Qwen0.5B. At the 3B mid-scale, *BMT* reaches $81.49/82.49 \approx 99\%$ on Llama3B and $41.98/44.35 \approx 95\%$ on Qwen3B. At the largest backbone in each family, *BMT fully recovers* FPT performance: 83.55% vs. FPT 83.24% on Llama8B GSM8K, and 52.32% vs. FPT 51.89% on Qwen7B SimpleRL-Zoo. This scale-dependent pattern aligns with Gan & Isola (2026): as backbones grow, task-expert solutions densely populate the neighborhood of pretrained weights, so even the discrete set of binary keep/zero configurations intersects this expert region, while at smaller backbones the expert region is sparser, leaving floating-point updates an advantage that discrete keep/zero choices lack. §5.5 reinforces this: on *BMT* and FPT runs we replicate the spectrum-preserving, off-principal RLVR signature of Zhu et al. (2025) and additionally observe low effective rank, consistent with fine-tuning at scale operating near subset selection over a dense task-expert neighborhood.

Competitive with every baseline at the largest backbones. At Llama8B and Qwen7B, *BMT* matches or exceeds FPT, both LoRA ranks, and both MTH sparsity levels, with every method falling within noise across the two settings. Notably, all methods improve substantially over the base model (60.27% to $\sim 83\%$ on Llama8B GSM8K, 40.18% to $\sim 52\%$ on Qwen7B SimpleRL-Zoo). *BMT* thus departs from prior sparsity methods that aim to preserve base performance under aggressive pruning (Fang et al., 2024; Sun et al., 2024; Frantar & Alistarh, 2023): here, sparsity is the learning mechanism itself, matching strong fine-tuning baselines rather than merely holding the base.

Smallest adapter at every scale. *BMT*'s adapter is 0.5 MB at Qwen0.5B and 8.4 MB at Llama8B, several 1,000× smaller than the full-parameter 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 (the highest MTH sparsity that remained stable in preliminary experiments). Both are $\sim$ 1 to 4× larger than *BMT* across the six runs, and the denser 99% MTH setting is $\sim$ 50–60× larger. *BMT* thus carries the smallest information footprint of any adapter family: well under 0.1 bit per target weight after compression, while every other method writes floating-point deltas at modified positions.

5.3. Learning matters: Wanda alone barely moves the base performance

BMT substantially lifts every base model despite zeroing fewer than 0.05% of weights (Table 1). Does the Wanda importance prior we use to initialize the gates already do most of the work, with GRPO contributing only marginal refinement? We test this directly: for each run in Table 1, we apply Wanda one-shot pruning without GRPO training, sweep 8 sparsity levels from $1/2$ to $1/8192$, evaluate on the same validation set, and record the *best* accuracy across the sweep (full level list and per-sparsity breakdown in Appendix F). Table 2 compares this best-Wanda score against the base model and the trained *BMT* mask.

The Wanda row never moves more than 2.4% from the base, while *BMT*, starting from the same Wanda-prior init (Eq. (9)) but training the gates under GRPO, lifts every model by up to 53% (Llama gains 23–53%). The gain therefore comes from learning *which* sub-percent of weights to delete, not from the handcrafted importance signal alone.

Table 2. **Wanda alone only preserves base performance, while the gains in Table 1 come from learning the gates.** Best validation accuracy of Wanda one-shot pruning across sparsity sweeps, the base model, and the trained *BMT* mask.

	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 η)	1.59	52.46	60.65	11.42	38.70	42.49
BMT	**54.44**	**81.49**	**83.55**	**17.64**	**41.98**	**52.32**

5.4. BMT forgets less than FPT

Shenfeld et al. (2026b) establish that forward KL between the fine-tuned and base policies on the new task is a strong predictor of prior-task forgetting. We hypothesize that *BMT*'s very low learned sparsity (<0.05% of weights, Table 7) keeps the perturbation to the base policy small, yielding low forward KL and therefore little forgetting. Following Shenfeld et al. (2026b)'s evaluation, Figure 2 reports forward KL and prior-task accuracy on a six-task suite (HellaSwag, MMLU, TruthfulQA, Winogrande, IFEval, HumanEval) along the training trajectory for FPT and *BMT* at the three Llama scales.

FPT's forward KL grows monotonically to 1.0 on Llama1B and 1.5 on Llama8B by step 348, with prior-task accuracy correspondingly dropping 1–3% below base. In contrast, *BMT*'s forward KL plateaus at ≤ 0.05 across every backbone, and prior-task accuracy holds at or slightly above base throughout training. *BMT* thus delivers the GSM8K gains in Table 1 without measurably eroding base capabilities.

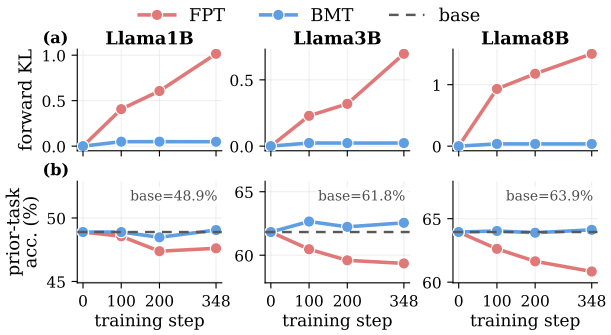

Figure 2. ***BMT* preserves prior-task accuracy across Llama scales while FPT drifts.** (a) Forward KL between the trained and base policies on the new task, the forgetting predictor of Shenfeld et al. (2026b). (b) Mean accuracy on a six-task prior-task retention suite. Dashed line marks the base reference.

5.5. Why BMT works

How can binary masking alone (§5.2) match full-parameter GRPO at scale? We characterize the learning dynamics through three update-geometry measures of the saved checkpoints to find out.

Update geometry. Let W_0 denote the pretrained value of a weight matrix, W_+ its value at the saved checkpoint (Llama8B step 348, Qwen7B step 100), and $\Delta W = W_+ - W_0$ the weight delta. Three quantities characterize how ΔW relates to W_0, averaged over all attention and MLP linear layers:

- **Effective rank** is defined as the stable rank of ΔW: $\|\Delta W\|_F^2/\|\Delta W\|_2^2 = (\sigma_1^2 + \cdots + \sigma_r^2)/\sigma_1^2$, where $\sigma_1 \geq \cdots \geq \sigma_r$ are ΔW's singular values in decreasing order. If a single singular value dominates the rest, the ratio is near 1; if k singular values are similar in size, the ratio is near k. It counts how many independent directions the update actually uses.
- **Spectrum drift** measures how much the weight matrix's singular-value magnitudes changed during training, relative

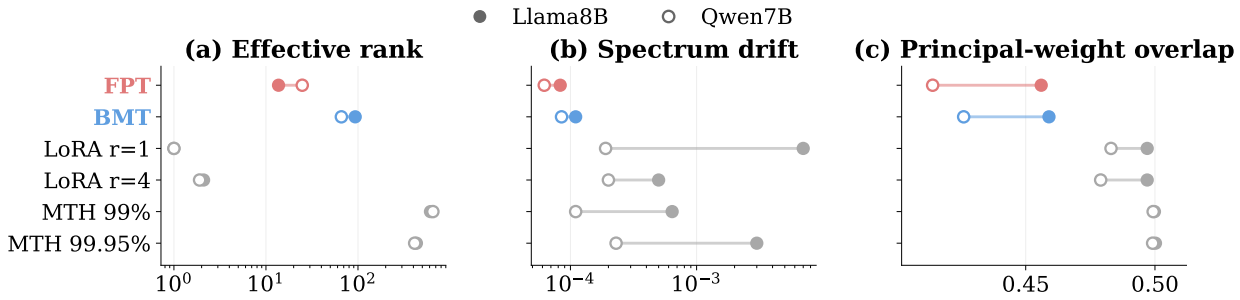

Figure 3. **Update geometry: *BMT* (blue) lands closest to FPT (red) on every axis. LoRA and MTH (gray) sit elsewhere.** Each panel shows one geometric property at the saved checkpoint: (a) effective rank of ΔW, (b) spectrum drift, (c) principal-weight overlap. Sub-random overlap (below the random baseline of 0.5) is the RLVR signature of Zhu et al. (2025).

Table 3. **SFT on Llama.** Best-checkpoint evaluation accuracy (%) and adapter size (MB) on MMLU-Pro and BBEH at 1B and 3B.

	MMLU-Pro				BBEH			
	1B		3B		1B		3B	
Method	**Acc. (%)**	**Size (MB)**	**Acc. (%)**	**Size (MB)**	**Acc. (%)**	**Size (MB)**	**Acc. (%)**	**Size (MB)**
Base	10.37	0	12.63	0	2.17	0	8.87	0
FPT	16.97	4714	18.35	12256	**8.35**	4714	**11.39**	12256
BMT	**18.62**	**1.0**	**22.28**	**3.1**	5.65	**1.0**	**11.39**	**3.1**

Table 4. **SDFT on Qwen7B, ToolAlpaca tool-use.** Best-checkpoint exact-match accuracy and adapter size (MB).

Method	**Acc. (%)**	**Size (MB)**
Base	41.24	0
FPT	68.04	29051
BMT	**69.07**	**6.9**

to where they started: $\|\sigma(W_+) - \sigma(W_0)\|_2 / \|\sigma(W_0)\|_2$ (Zhu et al., 2025), where $\sigma(W)$ is the vector of singular values of W. Small drift means the singular-value magnitudes are nearly the same before and after; large drift means they changed substantially.

- **Principal-weight overlap** measures whether the update lands on what Zhu et al. (2025) call the base model's *principal weights*: the entries with the highest magnitude in the rank-128 truncated SVD of W_0. Define two binary masks of W_0's shape: (i) the *update mask* M_{update}, with 1 at positions where $|W_+(i,j) - W_0(i,j)| > 10^{-3} \cdot \max(|W_0(i,j)|, |W_+(i,j)|)$, a relative-tolerance test from Zhu et al. that filters out changes too small to register a different bf16-stored value, and (ii) the *principal-weight mask* $M_{\text{princ}} = \text{Top}_{50\%}(|W_0^{(128)}|)$, the top 50% entries by absolute value of $W_0^{(128)}$, the best rank-128 approximation of W_0. The metric is the overlap fraction $|M_{\text{princ}} \cap M_{\text{update}}| / |M_{\text{update}}|$. The random baseline is 0.5 (the density of M_{princ}): if update positions were uniformly distributed, half would land on principal weights. Sub-random overlap (below 0.5) is the RLVR signature reported by Zhu et al. (2025).

Figure 3 reports all three metrics across the six fine-tuning methods. ***BMT* lands closest to FPT on every axis.** Its effective rank (66–93) is much closer to FPT's (14–25) than to MTH's (400–650) or LoRA's structurally rank-deficient 1–2. Its spectrum drift sits within 1.4× of FPT, while LoRA r=1 exceeds FPT by up to 84× on Llama8B. Its principal-weight overlap of 0.43–0.46 matches FPT's sub-random pattern of 0.41–0.46, whereas LoRA stays close to the random baseline of 0.5 (0.48–0.50) and MTH sits on it.

Interpretation. Spectrum preservation and the sub-random principal-weight overlap are the RLVR signature reported by Zhu et al. (2025), while the low effective rank we measure for FPT is, to the best of our knowledge, a novel empirical observation. Of the four adapter families we compare, *BMT* is the only one that shares two properties with FPT: the rank of ΔW is unconstrained (whereas LoRA caps it at r by construction), and the choice of which weights to update is itself learned (whereas MTH fixes a random mask at initialization and never revises it). We believe this shared freedom over rank and weight selection is why *BMT*'s update geometry tracks FPT's at scale (Table 1), even though no weight is ever updated to a new non-zero value.

5.6. Generalizing beyond GRPO: results on SFT and SDFT

Does the same parameterization work outside on-policy RL? We test *BMT* on two other widely used fine-tuning regimes: standard off-policy supervised fine-tuning (SFT) on multi-task knowledge and reasoning benchmarks, and on-policy

Table 5. ***BMT*** **hyperparameter sensitivity: Llama8B on GSM8K (%).** Each cell is best-checkpoint accuracy at each (η, λ, 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 6. ***BMT*** **hyperparameter sensitivity: Qwen7B on SimpleRL-Zoo (%).** Each cell is best-checkpoint accuracy at each (η, λ, 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

distillation (SDFT (Shenfeld et al., 2026a)) on a tool-use task. Both reuse the gate-projection target and the Wanda-prior logit initialization of §4. For SFT, we fine-tune Llama at 1B and 3B separately on each of the 14 MMLU-Pro (Wang et al., 2024) subjects and 23 BBEH (Kazemi et al., 2025) subtasks, and report the mean test accuracy across tasks within each suite (Appendix E for per-task counts). For SDFT, we follow Shenfeld et al. (2026a) on ToolAlpaca with Qwen7B as both the student and the EMA teacher, using their held-out tool-call test split and regex evaluation protocol verbatim.

SFT. *BMT* matches or exceeds FPT on three of four MMLU-Pro and BBEH runs under a single SFT configuration shared across all subjects and subtasks, with an adapter several $1{,}000\times$ smaller (Table 3). It exceeds FPT on both MMLU-Pro runs (Llama1B 18.62 vs. 16.97 and Llama3B 22.28 vs. 18.35), ties on BBEH 3B at 11.39, and trails on BBEH 1B (5.65 vs. 8.35).

SDFT. *BMT* matches FPT on Qwen7B ToolAlpaca tool-use (69.07% vs. 68.04%, Table 4) with a 6.9 MB adapter, several $1{,}000\times$ smaller than the FPT checkpoint. SDFT's setup naturally suits *BMT*: on-policy student rollouts give a clean gradient signal on the gates, and the slow EMA teacher (which lags the student and stays close to the base over short runs) provides a soft form of the retention we measure for GRPO in §5.4. The binary-mask parameterization stays close to the base by construction, reinforcing this anchor.

5.7. BMT is robust to hyperparameter changes

BMT is robust to its hyperparameters: initial sparsity η, learning rate lr, and sparse-weight regularization strength λ. 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 5 and 6). Every run falls within $\pm 1.5\%$ of the best (83.55%) on Llama8B and within $\pm 3.2\%$ of the best (52.80%) on Qwen7B, so the headline numbers in Table 1 are stable to the choices we make in this region.

6. Conclusion

We introduced *Bit-Mask Tuning* (*BMT*), a fine-tuning parameterization whose entire learned object is a binary keep/zero mask over the gate projection of transformer blocks. At the top-scale run in each regime (Llama8B GSM8K, Qwen7B SimpleRL-Zoo, Qwen7B ToolAlpaca, Llama3B MMLU-Pro and BBEH), *BMT* matches full-parameter fine-tuning across GRPO, SDFT, and SFT, and the GRPO gap to FPT contracts monotonically as the backbone grows.

Beyond matching FPT in accuracy, two practical consequences distinguish *BMT*. The trained mask falls below even the smallest possible LoRA (rank 1) on adapter size, three orders of magnitude under the full-parameter checkpoint. *BMT* also forgets substantially less than FPT: on our GRPO retention runs, it holds prior-task accuracy throughout training while FPT consistently loses 1–3%.

To understand why masking alone matches FPT, we study the learning dynamics through update-geometry analysis on GRPO. Every adapter we evaluate (LoRA at two ranks, MTH at two sparsity levels) drifts from full-parameter updates in distinct ways, while *BMT*'s weight delta lands closest to FPT on effective rank, spectrum drift, and principal-weight overlap. Among adapters, masking tracks FPT's update geometry most closely.

Taken together, the matched accuracy, smaller adapter, reduced forgetting, and update geometry closer to FPT's than that of any other adapter suggest a different implicit-regularization framing of fine-tuning at scale: rather than writing new values into a pretrained model, it may suffice to choose which existing values to delete. More fundamentally, that the gap closes with model capacity suggests fine-tuning at scale is selection within a dense task-expert neighborhood of pretrained weights, not synthesis of new parameter values.

References

Adewuyi, I., Okibe, S., and Ivanov, V. The multiple ticket hypothesis: Random sparse subnetworks suffice for RLVR. *arXiv preprint arXiv:2602.01599*, 2026.

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

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. Sparse high rank adapters. In *NeurIPS*, 2024.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Collet, Y. and Kucherawy, M. Zstandard compression and the `application/zstd` media type, 2018. IETF RFC 8478.

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

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

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

Gan, Y. and Isola, P. Neural thickets: Diverse task experts are dense around pretrained weights. *arXiv preprint arXiv:2603.12228*, 2026.

Grattafiori, A., Dubey, A., Jauhri, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

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

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

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

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

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

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. BIG-bench extra hard. In *ACL*, 2025.

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. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

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

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

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

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

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

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

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

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

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. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

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

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

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

Shenfeld, I., Damani, M., Hübotter, J., and Agrawal, P. Self-distillation enables continual learning. In *International Conference on Machine Learning*, 2026a.

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

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

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *ICLR*, 2025.

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

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

Sun, Q., Cetin, E., and Tang, Y. Transformer^2: Self-adaptive LLMs. In *ICLR*, 2025.

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

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

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. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS Datasets and Benchmarks*, 2024.

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

Yang, A., Yang, B., Zhang, B., et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

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

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

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. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 2023.

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

Zhu, H., Zhang, Z., Huang, H., Su, D., Liu, Z., Zhao, J., Fedorov, I., Pirsiavash, H., Sha, Z., Lee, J., Pan, D. Z., Wang, Z., Tian, Y., and Tai, K. S. The path not taken: RLVR provably learns off the principals. In *NeurIPS Workshop on Efficient Reasoning*, 2025.

Zhu, M. and Gupta, S. To prune, or not to prune: Exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A. Limitations and future work

BMT's strength compounds with model scale (Table 1), and we hope to test it on backbones beyond the 7–8B regime we evaluate, as well as on mixture-of-experts (MoE) architectures, where the gate projection plays a different architectural role. A theoretical account of why masking alone matches FPT remains open and would complement our geometric analysis (§5.5). Other promising directions include sweeping which target modules to sparsify, exploring structured row- and column-pruning variants, designing curriculum-driven gate schedules, and applying *BMT* to broader fine-tuning regimes.

B. Adapter size

Each Size (MB) in Tables 1, 3, and 4 is a lower bound on the bytes-on-disk needed to reconstruct the trained policy from the public base model checkpoint, ensuring a fair comparison across methods. We exclude training-framework overhead discarded at deployment (safetensors/HF/PEFT headers, optimizer states like Adam m/v, training-resume buffers like LR-scheduler and RNG state), which easily becomes significant in comparison to the meaningful payload at small adapter sizes.

Full-parameter fine-tuning (FPT). A fine-tuned LLM checkpoint at fp32:

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

where N_{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:

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

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

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

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

BMT. A compressed binary keep/zero mask over the masked weights:

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

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 of §5.5, zstd compresses to roughly 0.03 bits per masked weight.

C. Learned BMT sparsity per run

Table 7 reports learned total-network sparsity for the GRPO runs of Table 1 ($\eta = 1/2048$, $\lambda = 10^{-5}$, $lr = 3{\times}10^{-4}$) at the earliest and final saved checkpoints. The denominator covers all model parameters. Sparsity grows from earliest to final in every run, with all values well below the 0.05% ceiling.

Table 7. **Learned *BMT* total-network sparsity for the six GRPO runs.** Fraction of all model parameters set to zero by the trained mask, at the earliest and final saved checkpoint. All values fall well below the 0.05% ceiling.

	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%

D. 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. Each block covers one Qwen2.5 model size. Columns are the six SimpleRL-Zoo evaluation benchmarks.

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						
FPT	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
BMT	44.12	33.40	5.51	6.98	3.33	12.50
Qwen3B						
FPT	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
BMT	83.78	66.20	22.79	25.78	13.33	40.00
Qwen7B						
FPT	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
BMT	91.28	78.80	32.72	40.29	23.33	47.50

E. 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

F. Full Wanda sparsity sweep

Tables 11 and 12 report the per-sparsity breakdown underlying §5.3's best-Wanda picks. η is the fraction of gate-projection weights deleted (same convention as Eq. (9)), 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 η), while the trained *BMT* runs of Table 1 use the noisy init of Eq. (9). Qwen values are the six-benchmark mean across GSM8K, MATH500, Minerva, OlympiadBench, AIME24, and AMC23. "Base" is no fine-tuning. Bold marks the best Wanda accuracy per backbone.

Table 11. **Wanda one-shot pruning sweep, Llama on GSM8K (%).**

η	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 one-shot pruning sweep, Qwen2.5 on SimpleRL-Zoo six-benchmark mean (%).**

η	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

G. Algebraic connection to MaskLLM

BMT and MaskLLM (Fang et al., 2024) arrive at their forward passes from different starting points: *BMT* from a continuous relaxation of a per-weight keep/zero decision (Eq. (6)), and MaskLLM from a Gumbel-Softmax over $N{:}M$ candidate patterns (Eq. (10)). After three structural modifications to Eq. (10) (binary candidates, no per-group cardinality, no Gumbel noise), the modified MaskLLM forward matches Eq. (6) 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 the $C(M,N) = \binom{M}{N}$ candidate binary patterns $\{\widehat{M}_c\}_{c=1}^{C(M,N)}$ that keep exactly N of M entries, 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):

$$\widetilde{y}_c \;=\; \frac{\exp\big((\kappa\,\pi_c + \gamma_c)/\tau\big)}{\sum_{c'} \exp\big((\kappa\,\pi_{c'} + \gamma_{c'})/\tau\big)}, \qquad \gamma_c \overset{\text{iid}}{\sim} \text{Gumbel}(0,1). \tag{10}$$

The group's soft mask is the candidate-pattern-weighted average $\widetilde{M} = \sum_c \widetilde{y}_c \widehat{M}_c$, 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 τ and logit-scale κ both follow linear schedules over training.

Step 1: $C(M,N){=}2$ collapses softmax-of-2 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:

$$\widetilde{y}_0 = \frac{e^{(\kappa\pi_0+\gamma_0)/\tau}}{e^{(\kappa\pi_0+\gamma_0)/\tau} + e^{(\kappa\pi_1+\gamma_1)/\tau}} = \frac{1}{1 + e^{-\frac{1}{\tau}[\kappa(\pi_0-\pi_1)+(\gamma_0-\gamma_1)]}} = \sigma\Big(\tfrac{1}{\tau}\big[\kappa\,(\pi_0-\pi_1) + (\gamma_0-\gamma_1)\big]\Big), \tag{11}$$

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 $\text{Gumbel}(0,1)$ variables is $\text{Logistic}(0,1)$, so

$$\widetilde{y}_0 \;=\; \sigma\big(\tfrac{1}{\tau}[\kappa\,\Delta\pi + \xi]\big), \qquad \Delta\pi := \pi_0 - \pi_1,\;\; \xi \sim \text{Logistic}(0,1). \tag{12}$$

Eq. (12) 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 independently decide keep/zero replaces this group categorical with an independent binary keep/zero choice per weight, parameterized by a single logit G_{ij}. Each weight then has its own Binary-Concrete (Eq. (12) applied per-weight, with G_{ij} in place of $\Delta\pi$):

$$\widetilde{M}_{ij} \;=\; \sigma\big(\tfrac{1}{\tau}(\kappa\,G_{ij} + \xi_{ij})\big), \qquad \xi_{ij} \sim \text{Logistic}(0,1). \tag{13}$$

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

$$\widetilde{M}_{ij} \;=\; \sigma\big(\tfrac{\kappa}{\tau}\,G_{ij}\big). \tag{14}$$

This *matches* Eq. (6) with $\alpha(t) := \kappa(t)/\tau(t)$, which is a scheduled scalar (not linear in t, since κ and τ are scheduled individually).