BEYOND INSTANTANEOUS GRADIENTS: MOMENTUM-GUIDED ADAPTIVE GRADIENT MASKING FOR SPARSE FINE-TUNING OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse fine-tuning of large language models (LLMs) requires selecting the parameters that contribute most to downstream adaptation under a limited update budget. Existing gradient-based selection methods rely primarily on gradient statistics from the current optimization stage, making their importance estimates susceptible to mini-batch noise, outliers, and short-term fluctuations in optimization. We propose Momentum-Guided Adaptive Gradient Masking (MAGM) to address this issue. MAGM uses an exponential moving average to integrate gradient information across training steps, yielding a history-aware importance estimate that captures sustained parameter contributions. These importance scores guide layer-wise budget allocation and within-layer parameter selection under a fixed global update budget. Experiments show that MAGM achieves the highest average scores among the compared methods in code generation, mathematical reasoning, and general-domain evaluation. On DeepSeek-Coder-Base-6.7B and MISTRAL-7B, MAGM improves the average code-generation score by 1.63% and 0.52%, respectively, over the best-performing algorithm. In mathematical reasoning, MAGM achieves average performance comparable to the best-performing baseline on both MISTRAL-7B and LLAMA3-8B. On general-domain tasks, the gains over the best-performing algorithm are 2.0% for supervised fine-tuning and 2.1% for direct preference optimization. These results indicate that MAGM enables more stable parameter importance estimation and more effective budget allocation under a limited parameter update budget.

1 INTRODUCTION

Large language models (LLMs) have achieved strong performance in natural language understanding, code generation, and complex reasoning. As these models grow in size, however, updating all parameters during fine-tuning requires substantial computation and memory, while also increasing the costs of deployment and transfer to new tasks. Updating fewer parameters without compromising performance is an important goal in efficient model adaptation (Albert et al., 2025).

Parameter-efficient fine-tuning (PEFT) adapts a model to downstream tasks at a lower resource cost by updating only a subset of its parameters or introducing a small number of trainable parameters (Liu et al., 2025). Existing approaches either introduce additional trainable parameters or select existing parameters for updating. For example, low-rank adaptation reduces the number of trainable parameters by learning low-rank update matrices (Wang et al., 2025b), while approaches that combine quantization and low-rank adaptation further reduce adaptation costs (Zhou et al., 2025). These methods nevertheless introduce additional structure, which can increase model complexity. Parameter-selection methods instead operate directly in the pretrained parameter space, identifying task-relevant parameters for sparse updates to reduce computational overhead without introducing additional modules.

Gradient-based selection uses parameter gradients to estimate importance for task optimization and selects a subset of parameters for updating. Gradient-Mask Tuning (GMT) (Li et al., 2025), for example, accumulates gradients over multiple mini-batches and constructs a parameter-selection mask from gradient magnitudes, enabling model adaptation under a limited update budget. Exist-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ing gradient-masking methods, however, base their selection primarily on the currently accumulated gradients. Their importance estimates reflect the current optimization state without explicitly retaining information from earlier updates. Moreover, methods that use a fixed update ratio or global parameter-level selection determine the update set from parameter importance without explicitly modeling differences in contribution across Transformer layers.

Prior work on PEFT also shows that the expressiveness of the update space directly affects downstream adaptation (Albert et al., 2025). Parameter importance is not determined solely by the gradient at a single optimization stage; it develops over the course of training. Some parameters continue to influence optimization across multiple stages, and current gradients alone may not adequately capture these sustained contributions. Gradient history and model structure provide complementary information for improving importance estimation and allocating sparse updates.

MAGM performs sparse parameter updates through momentum-guided importance estimation and layer-adaptive budget allocation. An exponential moving average combines current gradient magnitudes with historical information to reduce the influence of short-term gradient fluctuations on importance estimation. MAGM aggregates parameter importance within each layer to determine its update share, then selects parameters within the allocated budget.

Our main contributions are as follows:

- We introduce a momentum-guided parameter importance estimator that maintains a history of gradient-based importance for each parameter, improving selection stability compared with relying solely on current gradient magnitudes.
- We propose a layer-adaptive mask ratio allocation strategy that distributes the global update budget according to layer importance and selects parameters within each allocated budget.
- We evaluate MAGM on multiple downstream tasks. Under matched parameter update budgets, MAGM outperforms existing gradient-masking methods, demonstrating its effectiveness for sparse fine-tuning of LLMs.

2 Related Work

We review efficient LLM fine-tuning through parameter-efficient adaptation, sparse parameter updates, and structure-aware allocation.

Parameter-efficient adaptation. DoRA (Liu et al., 2024), LoRA-Pro (Wang et al., 2025b), EVA (Paischer et al., 2025), and KaSA (Wang et al., 2025a) improve low-rank adaptation through changes to parameterization, optimization, or initialization. LoFiT (Yin et al., 2024) instead learns offset vectors for representations at selected attention heads. These methods improve adaptation through compact trainable parameterizations or representation interventions. Their focus on compact adaptation spaces leaves open the question of which original pretrained parameters to update under a sparse budget.

Sparse parameter updates. S^2FT (Yang et al., 2024) updates selected attention and feed-forward structures, while SMT (He et al., 2025) selects sparse submatrices for fine-tuning. Other approaches exploit transferable parameter sensitivity (Guo et al., 2025) or salience-based sparse adaptations (Liu et al., 2025). Closely related to our work, GMT (Li et al., 2025) selects parameters using gradient magnitudes accumulated over mini-batches within each update interval. Its selection criterion does not explicitly maintain an importance estimate across optimization steps, leaving parameter selection sensitive to short-term gradient fluctuations. This motivates incorporating gradient history into the selection criterion.

Structure-aware allocation. LoSA (Huang et al., 2025) combines layer-wise sparsity allocation with adaptive low-rank recovery. IGU-LoRA (Cui et al., 2026), RaLoRA (Ye et al., 2026), and FlexLoRA (Liu et al., 2026) adapt low-rank capacity using sensitivity, gradient dimensionality, or spectral information. In particular, IGU-LoRA uses exponential moving averages and deviation tracking to stabilize importance estimates. Layer-pruning studies (Gromov et al., 2025; Ding et al., 2026) provide complementary evidence of structural redundancy through pruning. Although these studies establish the value of accounting for layer differences, their allocation and pruning mech-

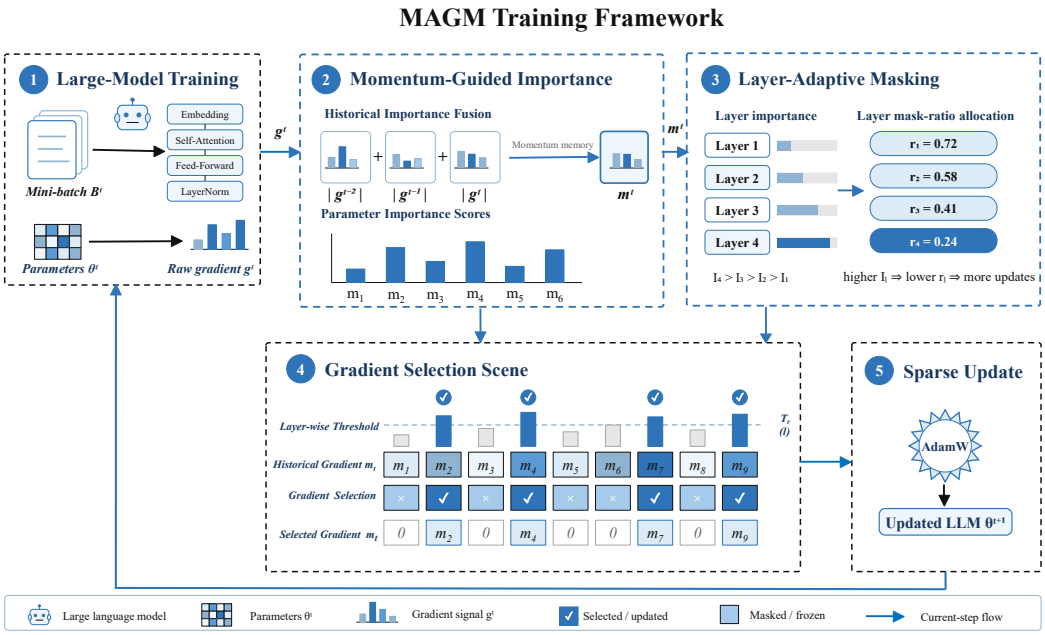


Figure 1: Overview of MAGM. Importance scores guide layer-wise budget allocation and parameter selection. The resulting mask is applied to current gradients for sparse updates.

anisms primarily concern adaptation capacity or model compression. Distributing a sparse update budget across the original pretrained parameters remains a separate problem.

MAGM accounts for gradient fluctuations and differences in layer importance when selecting pretrained parameters for sparse updates. It estimates importance across training steps, allocates a fixed update budget according to layer importance, and selects parameters within each layer.

3 Methodology

3.1 Overview

Figure 1 illustrates how MAGM converts gradient observations into sparse updates through importance estimation, layer-wise budget allocation, and parameter selection. The following sections formalize these operations.

MAGM begins by computing gradients over consecutive mini-batches to obtain parameter importance observations. It then combines current gradient contributions with historical importance estimates through an exponential moving average, producing history-aware importance scores for individual parameters. Aggregating these scores yields a layer importance estimate for allocating mask ratios under the global update budget. The assigned ratio determines the threshold used to select parameters within each layer. The optimizer then updates the selected parameters using the current gradients while keeping the remaining parameters frozen. This procedure reduces the influence of short-term gradient fluctuations on parameter selection while adapting update budgets to layer importance.

3.2 Momentum-Guided Parameter Importance Estimation

MAGM constructs parameter-level importance observations from consecutive mini-batches and smooths them across updates. For each training mini-batch, we compute the gradients of the loss $\mathcal{L}$ with respect to the model parameters. The gradient accumulation interval N is the number of mini-batches processed before each parameter update. Let $\mathcal{B}_n$ denote the nth mini-batch, and let

$\nabla_{\theta_{ij}}\mathcal{L}(\theta, \mathcal{B}_n)$ denote the gradient of its loss with respect to parameter θ_{ij}. The accumulated gradient for each parameter θ_{ij}, denoted by Γ_{ij}, is defined as

$$\Gamma_{ij} = \frac{1}{N}\sum_{n=1}^{N} \nabla_{\theta_{ij}}\mathcal{L}(\theta, \mathcal{B}_n) \tag{1}$$

The parameter-selection criterion used by GMT (Li et al., 2025) is

$$S_{ij}^{\mathrm{GMT}} = |g_{ij}| \tag{2}$$

where g_{ij} is the accumulated gradient for parameter θ_{ij}. Using only its current magnitude makes the importance estimate sensitive to gradient fluctuations. MAGM maintains an EMA buffer m_{ij}^t as the importance score, with g_{ij}^t denoting the accumulated gradient at update step t. The buffer is updated as

$$m_{ij}^t = \beta m_{ij}^{t-1} + (1-\beta)|g_{ij}^t| = (1-\beta)\sum_{k=0}^{t-1}\beta^k |g_{ij}^{t-k}| \tag{3}$$

Here, $m_{ij}^0 = 0$, g_{ij}^t is the accumulated gradient at update step t, and β controls the memory of earlier gradients. The contribution of a gradient magnitude observed k steps earlier decays with β^k. Appendix A provides the extended explanation.

3.3 Layer-Adaptive Update-Budget Allocation

Prior work on model sparsification shows that parameters differ substantially in their contributions to model outputs and that importance-based criteria can identify parameters with limited contributions (Le et al., 2025). Under a global mask ratio r_{global}, MAGM assigns each layer an adaptive mask ratio r_l based on its importance, while ensuring that the overall mask ratio matches the global target. We first compute an importance score for each layer l. Let P_l denote the parameter set of layer l, and let $|P_l|$ be the number of parameters in that layer. Its average importance is

$$imp_l = \frac{1}{|P_l|}\sum_{(i,j)\in P_l} m_{ij}^t \tag{4}$$

The layer score imp_l aggregates the parameter importance scores used for within-layer selection. We allocate mask ratios using a power-weighted rule controlled by the concentration parameter α:

$$r_l = 1 - \frac{imp_l^\alpha}{\sum_j imp_j^\alpha}\cdot(1-r_{\mathrm{global}})\cdot\frac{|P_{\mathrm{all}}|}{|P_l|} \tag{5}$$

Here, $|P_{all}|$ is the total number of parameters subject to masking, r_{global} is the global mask ratio, and r_l is the fraction of masked parameters in layer l. Each layer receives an update budget proportional to its importance raised to the power α. We constrain the initial layer-wise mask ratios through an iterative water-filling procedure under the global budget. Specifically, we first compute the unconstrained mask ratio for each layer from its importance. For any layer whose ratio falls outside the prescribed bounds, we fix the ratio at the corresponding boundary and exclude that layer from subsequent allocation. The remaining update budget is then redistributed among the other layers according to their importance. This process repeats until all layer-wise mask ratios lie within $[r_{min}, r_{max}]$ and the overall update ratio satisfies the global mask constraint:

$$\sum_l (1-r_l)|P_l| = (1-r_{\mathrm{global}})|P_{\mathrm{all}}|, \qquad r_{\min} \le r_l \le r_{\max} \tag{6}$$

The constraint ensures that the average update ratio, weighted by the number of parameters in each layer, matches the global target. Larger values of α concentrate more of the budget in layers with

Algorithm 1 Momentum-Guided Adaptive Gradient Masking (MAGM)

Require: Model parameters $\Theta^{(0)}$, learning rate η, momentum coefficient β, global mask ratio ρ, concentration coefficient α
Ensure: Updated parameters $\Theta^{(T)}$
Initialize momentum importance $M^{(0)} \leftarrow 0$
for $t = 1$ to T **do**
  Compute accumulated gradient $\Gamma^{(t)}$
  $M^{(t)} \leftarrow \beta M^{(t-1)} + (1-\beta)|\Gamma^{(t)}|$
  for each Transformer layer l **do**
    $I_l^{(t)} \leftarrow \mathrm{Mean}(M_l^{(t)})$
  end for
  Compute adaptive mask ratio $r_l^{(t)}$ according to $I_l^{(t)}$, α, and ρ
  Adjust $\{r_l^{(t)}\}$ by iteratively clipping out-of-bound mask ratios and redistributing the remaining update budget under the global constraint
  for each Transformer layer l **do**
    $\tau_l^{(t)} \leftarrow \mathrm{Percentile}(M_l^{(t)}, r_l^{(t)})$
    $S_l^{(t)} \leftarrow \mathbf{1}(M_l^{(t)} \geq \tau_l^{(t)})$
    $\Gamma_l^{(t)} \leftarrow S_l^{(t)} \odot \Gamma_l^{(t)}$
  end for
  $\Theta^{(t+1)} \leftarrow \Theta^{(t)} - \eta\Gamma^{(t)}$
end for

higher importance. To maintain robust updates in every layer, we impose lower and upper bounds r_{min} and r_{max} on its mask ratio, preventing a layer from being fully updated or fully frozen. For layer l, the allocated mask ratio r_l determines a threshold $T_r^{(l)}$ on the importance scores within that layer. Parameters are selected by

$$S_{ij}^{(l,t)} = \mathbf{1}\left[m_{ij}^t \geq T_r^{(l,t)}\right], \qquad T_r^{(l,t)} = \mathrm{Quantile}_{r_l}(m_l^t) \tag{7}$$

The binary mask selects the parameters to update. In gradient-descent form, the masked update is

$$\theta_{ij}^{(t+1)} = \theta_{ij}^{(t)} - \eta S_{ij}^{(l,t)} g_{ij}^t \tag{8}$$

where η is the learning rate and g_{ij}^t is the current accumulated gradient. Algorithm 1 summarizes mask construction using this illustrative gradient-descent update. The experiments use AdamW. Historical importance determines selection, while current gradients drive optimization.

Appendix A analyzes importance-estimation stability and layer-adaptive allocation. Under the stated approximately independent noise assumptions, the steady-state variance of the EMA estimate is scaled by $(1-\beta)/(1+\beta)$ relative to that of the current gradient-magnitude observation.

4 Experiments

4.1 Experimental Setup

We evaluate MAGM on code generation, mathematical reasoning, and general-domain tasks, with both SFT and DPO (see also Razin et al., 2025) in the general-domain setting.

All experiments use Hugging Face Transformers and PyTorch, bfloat16 precision, and RTX PRO 6000 GPUs. We use AdamW with a learning rate of 2×10^{-5}, cosine scheduling, and a warmup ratio of 0.03. Training begins with full-parameter updates to accumulate stable gradient information.

Table 1: Performance on code-generation and mathematical-reasoning benchmarks.

Method	Human Eval	Human Eval+	MBPP	MBPP+	Avg.	GSM8K	MATH	Avg.
	MISTRAL-7B					MISTRAL-7B		
Pre-trained	28	23.2	53.7	42.3	36.8	-	-	-
HFT	61.6	**58.9**	48.3	43.1	52.98	62.3	18.1	40.2
SFT	61	57.3	**50**	42.6	52.7	62.1	17.9	40.0
GMT	59.8	55.5	48.7	**43.4**	51.9	62.7	**18.5**	40.6
MAGM	**62.8**	57.9	**50**	**43.4**	**53.5**	**62.9**	18.4	**40.7**
	DEEPSEEK-CODER-BASE-6.7B					LLAMA3-8B		
Pre-trained	47.0	39.0	69.3	55.3	52.65	-	-	-
HFT	64.0	56.7	75.7	63.5	65.0	74.9	24.7	49.8
SFT	65.2	60.4	75.4	63	66	61.1	18.42	39.76
GMT	64.6	58.5	**76.2**	65.6	66.225	76.5	**25.74**	51.12
MAGM	**67.7**	**62.2**	75.4	**66.1**	**67.85**	**76.88**	25.4	**51.14**

4.1.1 BASELINES

We use the following baselines: **Full-parameter SFT** updates all trainable parameters (He et al., 2025). **HFT (Half Fine-Tuning)** (Hui et al., 2025) randomly selects a parameter subset for updating and freezes the rest. **GMT (Gradient-Mask Tuning)** (Li et al., 2025) selects parameters by gradient magnitudes accumulated within the current update interval. We match the update ratio across the sparse methods in each comparison and fix the random seed at 42.

4.1.2 TRAINING AND EVALUATION

Code generation. We fine-tune Mistral-7B and DeepSeek-Coder-Base-6.7B on Magicoder-Evol-Instruct-110K[1] (Wei et al., 2024) and evaluate on HumanEval, HumanEval+, MBPP, and MBPP+. The extended tests in HumanEval+ and MBPP+ are provided by EvalPlus[2] (see also Charakorn et al., 2025).

Mathematical reasoning. We fine-tune Mistral-7B and Llama3-8B on MetaMathQA (Yu et al., 2024) and evaluate on GSM8K and MATH, which cover different levels of mathematical difficulty.

General-domain tasks. We use TÜLU V2[3] (see also Jiang et al., 2025) for instruction tuning and UltraFeedback (Cui et al., 2024) for DPO. Evaluation on MMLU, GSM8K, BBH, TyDiQA, TruthfulQA, and HumanEval covers knowledge, mathematical and logical reasoning, multilingual question answering, truthfulness, and code generation.

4.2 MAIN RESULTS

4.2.1 CODE GENERATION

Table 1 compares MAGM with SFT, HFT, and GMT on Mistral-7B and DeepSeek-Coder-Base-6.7B.

MAGM achieves an average code-generation score of 67.85 on DeepSeek-Coder-Base-6.7B, exceeding GMT by 1.63 percentage points and full-parameter SFT by 1.85 points. On Mistral-7B, it also records the highest average, at 53.5 compared with GMT's 51.9. HumanEval and HumanEval+ show particularly clear improvements. The respective gains over GMT are 3.0 and 2.4 points on Mistral-7B and 3.1 and 3.7 points on DeepSeek-Coder-Base-6.7B. This pattern holds across both pretraining backgrounds and includes the extended evaluation in HumanEval+.

GMT selects parameters from gradient magnitudes within the current accumulation interval, leaving its importance estimates sensitive to local gradient fluctuations. MAGM integrates gradient infor-

[1] https://huggingface.co/datasets/ise-uiuc/Magicoder-Evol-Instruct-110K

[2] https://evalplus.github.io/leaderboard.html

[3] https://huggingface.co/datasets/allenai/tulu-v2-sft-mixture

Table 2: General-domain benchmark results after SFT on TÜLU V2 and DPO on UltraFeedback. Pretrained results are obtained from the corresponding base model. Bold denotes the best results.

Method	MMLU	GSM8K	BBH	TyDiQA	TruthfulQA	HumanEval	Avg.
Pre-trained	31.9	10	33.1	43.5	34.3	7.3	26.7
Supervised Fine-tuning (SFT) on TÜLU V2							
SFT	36.5	19.5	28.8	**52.0**	43.6	24.6	34.2
HFT	**37.9**	14.0	**34.2**	**52.0**	41.7	24.6	34.1
GMT	35.9	20.0	30.5	51.9	43.1	24.6	34.3
MAGM	36.9	**25.0**	33.4	**52.0**	**43.8**	**26.4**	**36.3**
Direct Preference Optimization (DPO) on UltraFeedback							
SFT	**39.3**	18.5	28.9	51.9	43.5	14.6	32.8
HFT	37.5	15.0	**33.8**	51.8	41.7	**17.7**	32.9
GMT	38.7	18.5	31.0	51.9	43.1	15.0	33.0
MAGM	39.2	**27.0**	33.6	**52.0**	**43.8**	15.2	**35.1**

All results use Llama2-7B. Evaluation uses MMLU, 0-shot EM; GSM8K, 8-shot CoT EM; BBH, 3-shot CoT EM; TyDiQA, 1-shot F1; TruthfulQA, 0-shot MC2; HumanEval, 0-shot Pass@10.

mation across successive training steps through an exponential moving average, producing more stable importance estimates that support consistent selection of parameters with sustained contributions to code adaptation. This stabilizing effect helps explain MAGM's improvements over GMT on the HumanEval benchmarks. Layer-adaptive allocation further directs the limited update budget toward layers with higher importance. Although the strongest individual results vary across methods, MAGM achieves the highest average score on both backbones, with clear gains over GMT on HumanEval and HumanEval+.

4.2.2 Mathematical Reasoning

Table 1 also reports mathematical-reasoning results for Mistral-7B and Llama3-8B.

On Llama3-8B, MAGM improves the average mathematical-reasoning score from 39.76 with full-parameter SFT to 51.14. Both benchmarks contribute to this gain, with GSM8K rising from 61.1 to 76.88 and MATH from 18.42 to 25.4. The Mistral-7B average increases from 40.0 to 40.7. Relative to GMT, MAGM records average gains of 0.02 points on Llama3-8B and 0.1 points on Mistral-7B. The comparison with full-parameter SFT is particularly favorable for Llama3-8B, where updating a selected subset yields better results on both benchmarks. On MATH, GMT leads by 0.1 points on Mistral-7B and 0.34 on Llama3-8B.

Reasoning examples produce different gradient contributions, making parameter selection based on the current accumulation interval sensitive to changes in the sampled problems. Historical smoothing reduces the influence of these fluctuations, while layer-adaptive allocation adjusts update shares to reflect differences in layer importance. Together, these operations concentrate limited updates on parameters with sustained contributions to reasoning adaptation, providing a mechanism for the gains over full-parameter SFT.

4.2.3 General-Domain Tasks

Table 2 reports general-domain performance after SFT on TÜLU V2 and DPO on UltraFeedback, covering the six benchmarks described in Section 4.1.2.

Supervised Fine-Tuning After instruction tuning, MAGM outperforms GMT on all six benchmarks, increasing the average score from 34.3 to 36.3. Gains of 5.0 points on GSM8K, 2.9 on BBH, and 1.8 on HumanEval accompany improvements in MMLU, TyDiQA, and TruthfulQA. The average also exceeds full-parameter SFT and HFT by 2.1 and 2.2 points. In a mixture of instruction-following tasks, each batch provides gradient information for only part of the training distribution. Selection based solely on that interval emphasizes its immediate contributions. MAGM carries information from earlier updates into the importance estimates, reducing the dependence of parameter

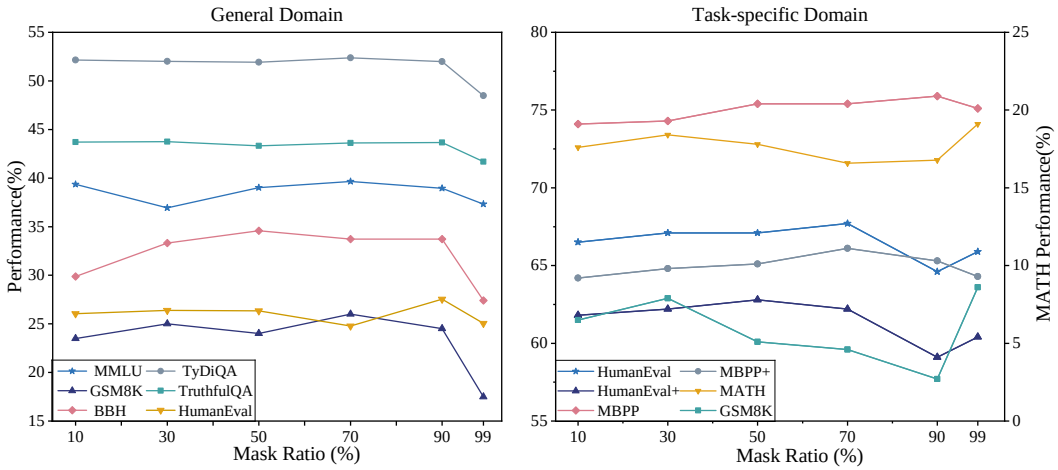


Figure 2: Effect of the masking ratio on downstream performance.

selection on short-term changes in batch composition. Layer-wise allocation uses these estimates to distribute capacity across the model, supporting adaptation to the instruction mixture and helping explain the gains across all six benchmarks.

Direct Preference Optimization The advantage persists under preference optimization. MAGM achieves an average score of 35.1, compared with 33.0 for GMT, and improves on all six benchmarks. GSM8K shows the largest gain, rising from 18.5 to 27.0, followed by BBH, which increases from 31.0 to 33.6. MAGM also records the highest TyDiQA and TruthfulQA scores among the compared methods. On HumanEval, MAGM scores 15.2 against HFT's 17.7. Here, preference gradients determine which parameters receive updates. MAGM incorporates new preference information while retaining earlier gradient contributions, so a parameter's importance is not determined by the current preference batch alone. The resulting scores guide both selection within layers and the distribution of updates across them. This combines responsiveness to the preference objective with continuity in importance estimation, helping account for the reasoning gains under a fixed update budget.

4.3 Ablation Study

4.3.1 Effect of the Mask Ratio

We evaluate mask ratios of 10%, 30%, 50%, 70%, 90%, and 99%. The mask ratio denotes the fraction of frozen parameters, so a larger value imposes a smaller update budget. TyDiQA and TruthfulQA remain relatively stable as the mask ratio increases from 10% to 90% (Figure 2), even though the updated fraction falls from 90% to 10%. MBPP also changes little over much of the tested range. Under a tighter budget, MAGM retains the highest-importance parameters within each layer and recalculates layer-wise update shares, concentrating the remaining capacity on parameters favored by the importance estimates. Historical smoothing reduces the influence of local gradient fluctuations on these rankings, which becomes especially relevant when fewer parameters can be selected. This combination explains how MAGM maintains useful adaptation on these benchmarks despite a substantial reduction in updated parameters. At 99% masking, only 1% of parameters remain available, and GSM8K and BBH decline more clearly as adaptation capacity becomes highly constrained.

4.3.2 Contributions of Individual Components

We compare the full method with two variants under otherwise unchanged training settings (Table 3).

MAGM-IGI (Instantaneous Gradient Importance) uses current gradient magnitudes for both layer-adaptive allocation and within-layer selection, replacing EMA-based importance scores.

Table 3: Ablation of momentum-guided importance estimation and layer-adaptive budget allocation.

Method	Human Eval	Human Eval+	MBPP	MBPP+	Avg.	GSM8K	MATH	Avg.
	Deepseek-CODER-BASE-6.7B					MISTRAL-7B		
MAGM-IGI	66.6	61.6	75.9	65.1	67.3	62.5	17.6	40.1
MAGM-UA	66.9	62.0	**76.5**	65.3	67.68	61.7	18.2	40.0
MAGM	**67.7**	**62.2**	75.4	**66.1**	**67.85**	**62.9**	18.4	**40.7**
GMT	64.6	58.5	76.2	65.6	66.225	62.7	**18.5**	40.6

Table 4: Training overhead under a matched experimental setup.

Method	Updated parameters (%)	GPU Memory	Time per Step
SFT	100%	43.9 GB	10.06s
GMT	30%	44.8 GB	10.49s
MAGM	30%	58.4 GB	10.64s

MAGM-UA retains EMA importance but selects parameters globally, without explicit layer-wise budget allocation.

The component comparisons in Table 3 show gains from both historical estimation and layer-wise allocation. Replacing instantaneous importance with gradient history raises the code-generation average from 67.3 to 67.85 and the mathematical-reasoning average from 40.1 to 40.7. MAGM-IGI determines importance from the current accumulation interval, making both parameter rankings and layer budgets responsive to local gradient fluctuations. Retaining historical contributions reduces this dependence and yields higher averages in both domains. MAGM also exceeds MAGM-UA by 0.17 points in code generation and 0.7 points in mathematical reasoning. Because MAGM-UA preserves historical estimation, this comparison isolates the value of explicit layer-wise allocation in addition to parameter ranking. Historical estimation contributes the larger gain in code generation, whereas the two components yield comparable gains in the mathematical-reasoning average. The higher averages of the complete method demonstrate the benefit of combining history-aware parameter ranking with explicit layer-wise allocation under the same update budget.

4.3.3 Training Overhead

We measure update ratios, GPU memory, and per-step training time under matched model, data, and hardware settings (Table 4), to assess optimization overhead (He et al., 2025).

With both methods updating 30% of parameters, MAGM takes 10.64 s per step and GMT takes 10.49 s (Table 4). The difference is 0.15 s, or approximately 1.4%. Importance tracking and layer-wise allocation operate on gradients already computed during training, keeping the additional step time modest in this configuration. Constructing the importance estimates requires no additional forward or backward pass through the model. The importance buffer adds persistent training state, increasing GPU memory from 44.8 to 58.4 GB, or approximately 30.4%. This configuration accommodates history-aware selection with a small increase in per-step runtime and additional storage for importance estimates.

5 Conclusion and Future Work

MAGM uses history-aware importance scores to allocate update budgets across layers and select parameters for sparse fine-tuning. It achieves higher average scores in code generation, mathematical reasoning, and general-domain evaluation, including both SFT and DPO. Its first-order estimator remains limited in how fully it captures changes in parameter contributions, and its applicability to architectures and training configurations beyond those evaluated here requires further study. Future work could incorporate second-order information or uncertainty estimates and adapt update strategies to different model sizes, tasks, and training stages.

AI USE STATEMENT

Generative AI tools were used to improve the clarity and readability of the manuscript. All AI-assisted textual content was reviewed by the authors, and the authors take full responsibility for the final content and claims presented in this work.

REFERENCES

P. Albert, F. Z. Zhang, H. Saratchandran, C. Rodriguez-Opazo, A. van den Hengel, and E. Abbasnejad. RandLoRA: Full Rank Parameter-Efficient Fine-Tuning of Large Models. In *International Conference on Learning Representations*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant Transformer Adaption. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 7485–7514, 2025.

G. Cui, L. Yuan, N. Ding, G. Yao, B. He, W. Zhu, Y. Ni, G. Xie, R. Xie, Y. Lin, Z. Liu, and M. Sun. ULTRAFEEDBACK: Boosting Language Models with Scaled AI Feedback. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 9722–9744, 2024.

X. Cui, H. Li, R. Zeng, Y. Zhao, J. Qian, W. Duan, B. Liu, and Z. Zhou. IGU-LoRA: Adaptive Rank Allocation via Integrated Gradients and Uncertainty-Aware Scoring. In *International Conference on Learning Representations*, 2026.

Xuan Ding, Pengyu Tong, Ranjie Duan, Yunjian Zhang, Rui Sun, and Yao Zhu. Pruning as a Cooperative Game: Surrogate-Assisted Layer Contribution Estimation for Large Language Models. In *International Conference on Learning Representations*, 2026.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The Unreasonable Ineffectiveness of the Deeper Layers. In *International Conference on Learning Representations*, 2025.

W. Guo, J. Long, Y. Zeng, Z. Liu, X. Yang, Y. Ran, J. Gardner, O. Bastani, C. De Sa, X. Yu, B. Chen, and Z. Xu. Zeroth-Order Fine-Tuning of LLMs with Transferable Static Sparsity. In *International Conference on Learning Representations*, 2025.

H. He, J. B. Li, X. Jiang, and H. Miller. SMT: Fine-Tuning Large Language Models with Sparse Matrices. In *International Conference on Learning Representations*, 2025.

W. Huang, Y. Zhang, X. Zheng, Y. Liu, J. Lin, Y. Yao, and R. Ji. Dynamic Low-Rank Sparse Adaptation for Large Language Models. In *International Conference on Learning Representations*, 2025.

Tingfeng Hui, Zhenyu Zhang, Shuohuan Wang, Weiran Xu, Yu Sun, and Hua Wu. HFT: Half Fine-Tuning for Large Language Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12791–12819, 2025.

Jinhao Jiang, Junyi Li, Wayne Xin Zhao, Yang Song, Tao Zhang, and Ji-Rong Wen. Mix-CPT: A Domain Adaptation Framework via Decoupling Knowledge Learning and Format Alignment. In *International Conference on Learning Representations*, 2025.

Qi Le, Enmao Diao, Ziyan Wang, Xinran Wang, Jie Ding, Li Yang, and Ali Anwar. Probe Pruning: Accelerating LLMs through Dynamic Pruning via Model-Probing. In *International Conference on Learning Representations*, 2025.

H. Li, X. Zhang, X. Liu, Y. Gong, Y. Wang, Q. Chen, and P. Cheng. Enhancing Large Language Model Performance with Gradient-Based Parameter Selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24431–24439, 2025. No. 23.

M. Liu, C. Si, and Y. Jia. FlexLoRA: Entropy-Guided Flexible Low-Rank Adaptation. In *International Conference on Learning Representations*, 2026.

S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. DoRA: Weight-Decomposed Low-Rank Adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32100–32121, 2024.

X. Liu, A. Thomas, C. Zhang, J. Cheng, Y. Zhao, and X. Gao. Refining Salience-Aware Sparse Fine-Tuning Strategies for Language Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31932–31945, 2025.

Fabian Paischer, Lukas Hauzenberger, Thomas Schmied, Benedikt Alkin, Marc Peter Deisenroth, and Sepp Hochreiter. Parameter Efficient Fine-tuning via Explained Variance Adaptation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. Unintentional Unalignment: Likelihood Displacement in Direct Preference Optimization. In *International Conference on Learning Representations*, 2025.

F. Wang, J. Jiang, C. Park, S. Kim, and J. Tang. KaSA: Knowledge-Aware Singular-Value Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2025a.

Z. Wang, J. Liang, R. He, Z. Wang, and T. Tan. LoRA-Pro: Are Low-Rank Adapters Properly Optimized? In *International Conference on Learning Representations*, 2025b.

Y. Wei, Z. Wang, J. Liu, Y. Ding, and L. Zhang. Magicoder: Empowering Code Generation with OSS-Instruct. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 52632–52657, 2024.

X. Yang, J. Leng, G. Guo, J. Zhao, R. Nakada, L. Zhang, H. Yao, and B. Chen. S^2FT: Efficient, Scalable and Generalizable LLM Fine-Tuning by Structured Sparsity. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

J. Ye, H. He, M. Li, F. Han, T. Chen, and P. Ye. Gradient Intrinsic Dimensionality Alignment: Bridging the Gap Between Low-Rank Adaptation and Full Fine-Tuning. In *International Conference on Learning Representations*, 2026.

F. Yin, X. Ye, and G. Durrett. LoFiT: Localized Fine-tuning on LLM Representations. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

L. Yu, W. Jiang, H. Shi, J. Yu, Z. Liu, Y. Zhang, J. T. Kwok, Z. Li, A. Weller, and W. Liu. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. In *International Conference on Learning Representations*, 2024.

Zikai Zhou, Qizheng Zhang, Hermann Kumbong, and Kunle Olukotun. LowRA: Accurate and Efficient LoRA Fine-Tuning of LLMs under 2 Bits. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 79570–79594, 2025.

A Additional Method Details and Theoretical Analysis

The momentum decay coefficient β controls how long gradient information is retained in the importance estimate. A gradient magnitude observed k steps earlier receives a weight proportional to β^k. In Eq. (3), a larger β produces slower decay, extending the influence of earlier observations. EMA smooths successive gradient measurements while continuing to incorporate information from the current optimization state.

A.1 Theoretical Analysis

We analyze the stability of importance estimation and the allocation of update resources across layers.

A.1.1 Stability of Momentum-Based Importance Estimation

Gradient magnitude is commonly used to measure parameter importance. However, random mini-batch sampling introduces training noise into instantaneous gradients, causing importance estimates to fluctuate. As described in Eq. (3), MAGM applies EMA to gradient history. Compared with using only the current gradient g_t, EMA integrates information from multiple training steps and reduces the influence of stochastic gradient noise on importance estimation. Let σ^2 denote the noise variance. We model the gradient-magnitude observation at training step t as

$$x^{(t)} = |g^{(t)}| = \mu + \varepsilon^{(t)} \tag{9}$$

where μ denotes the expected gradient magnitude and $\varepsilon^{(t)}$ represents perturbations introduced by random mini-batch sampling and other sources. We assume that the perturbations $\varepsilon^{(t)}$ are approximately independent and satisfy

$$\mathbb{E}[\varepsilon^{(t)}] = 0, \qquad \mathrm{Var}(\varepsilon^{(t)}) = \sigma^2 \tag{10}$$

Under the EMA importance update used by MAGM,

$$m^{(t)} = \beta m^{(t-1)} + (1-\beta)x^{(t)} \tag{11}$$

the noise variance at steady state is

$$\mathrm{Var}(m) = \frac{1-\beta}{1+\beta}\sigma^2 \tag{12}$$

For $0 < \beta < 1$, we have

$$\frac{1-\beta}{1+\beta} < 1 \tag{13}$$

Under these assumptions, the EMA importance estimate has lower noise variance than the current gradient magnitude. Increasing β strengthens the smoothing of historical observations, reducing the influence of local gradient perturbations on the estimated scores.

A.1.2 Layer-Adaptive Update Budget Allocation

Transformer layers serve different roles in representation learning and task adaptation, and their parameters contribute unevenly to optimization (Cui et al., 2026). Selecting parameters solely by global importance, without explicitly accounting for layer differences, may concentrate a limited update budget in only a few layers. MAGM aggregates parameter importance within layer l into $I_l = imp_l$. Before clipping, Eq. (5) allocates the number of selected parameters as

$$(1-r_l)|P_l| \propto I_l^{\alpha},$$

Here, α controls allocation concentration. For equal-sized layers with $\alpha > 0$ and neither ratio at a bound,

$$I_a > I_b \Rightarrow r_a < r_b.$$

The allocation preserves the global budget while assigning update shares according to layer importance. Within each layer, parameters are selected by their importance scores.

B Additional Discussion of Code Generation Results

HFT performs well on Mistral-7B, suggesting a benefit from restricting the set of updated parameters. On DeepSeek-Coder-Base-6.7B, GMT has the highest baseline average, supporting the use of task-specific gradients for parameter selection. MAGM exceeds both baselines in average performance on each model. Its history-aware estimates retain gradient information across updates and guide the selection of parameters for code adaptation.

At the benchmark level, MAGM performs well on HumanEval, HumanEval+, and MBPP+, with particularly clear gains on HumanEval. Relative to GMT, its HumanEval and HumanEval+ scores improve by 3.0 and 2.4 percentage points on Mistral-7B and by 3.1 and 3.7 percentage points on DeepSeek-Coder-Base-6.7B. GMT selects parameters primarily from the gradient magnitudes in the current accumulation interval, making its importance estimates sensitive to local fluctuations. MAGM instead integrates gradient information across training steps through EMA, producing more stable importance estimates and more consistently selecting parameters that contribute to code generation. The HumanEval results support the effectiveness of this selection strategy for code adaptation.

The highest average on DeepSeek-Coder-Base-6.7B shows that MAGM remains effective when the model has already undergone extensive code pretraining. Downstream adaptation still benefits from selective updates guided by the fine-tuning objective. Gradient history provides the importance estimates used to direct the limited budget toward relevant parameters.

The code generation results support the effectiveness of MAGM for domain-specific adaptation. Importance-based selection and adaptive allocation concentrate updates on parameters with sustained contributions to code adaptation.

C Additional Discussion of Mathematical Reasoning Results

Mathematical reasoning requires multiple deduction steps, and different training examples induce different gradient contributions. Importance estimates based only on current observations are sensitive to these variations. EMA integrates observations across updates to smooth parameter scores, while layer-adaptive allocation accounts for differences in their distribution across the model.

MAGM achieves the highest reported average scores across the evaluated code-generation and mathematical-reasoning settings. Compared with GMT, it combines gradient history with explicit layer-wise allocation to guide sparse updates. The results emphasize the role of parameter selection in fine-tuning, beyond the fraction of parameters updated.

D Additional Discussion of General-Domain Results

General-domain evaluation covers knowledge, mathematical and logical reasoning, multilingual question answering, truthfulness, and code generation. This setting tests adaptation across multiple capabilities under a limited update budget. We first perform SFT on TÜLU V2 and then apply DPO on UltraFeedback. Table 2 reports the results for both training paradigms.

MAGM achieves the best scores on GSM8K, TruthfulQA, and HumanEval, with the largest gains on GSM8K. Its GSM8K score of 25.0 exceeds the scores of SFT, HFT, and GMT by 5.5, 11.0, and 5.0 percentage points, respectively. On HumanEval, it reaches 26.4, improving over all three baselines by 1.8 percentage points. It also achieves the highest TruthfulQA score of 43.8. These results

demonstrate strong performance across mathematical reasoning, truthfulness, and code generation, with particularly pronounced gains on reasoning tasks that are sensitive to the quality of parameter updates.

HFT leads on MMLU and BBH by 1.0 and 0.8 points, respectively. MAGM nevertheless has the highest average across all six tasks, with its largest gains on reasoning and code generation.

The baseline comparisons further show that sparse fine-tuning performance depends on how parameters are selected, not only on how many are updated. HFT selects a fixed fraction of parameters and performs well on some tasks, but its selection lacks task-specific importance information, leading to variation across tasks. GMT introduces task information through current gradients, yet its importance estimates primarily reflect the current optimization stage. MAGM estimates importance from gradient history and adapts layer-wise update shares before selecting parameters within each layer.

MAGM achieves the highest scores on GSM8K, TyDiQA, and TruthfulQA. Its GSM8K score of 27.0 is 8.5 points higher than those of SFT and GMT and 12.0 points higher than that of HFT. This is the largest benchmark improvement in the DPO setting. This suggests that MAGM can preserve and update important reasoning-related parameters during preference optimization, making more effective use of a limited update budget. Its leading scores on TruthfulQA and TyDiQA also indicate effective adaptation for truthfulness and question answering.

MAGM is within 0.1 points of the best MMLU score and 0.2 points of the best BBH score, although HFT retains the higher HumanEval score. Its average of 35.1 remains the highest among the compared methods.

Across SFT and DPO, MAGM outperforms full-parameter SFT on average despite updating fewer parameters. The comparison with HFT supports task-informed selection, while the comparison with GMT favors incorporating gradient history and explicitly distributing the budget across layers. These advantages hold under both instruction and preference objectives.

The gains across reasoning, truthfulness, code generation, and question answering establish the breadth of MAGM's results in the evaluated multitask setting. History-aware importance estimation and layer-adaptive allocation provide a common approach to sparse adaptation under both training objectives.

E Additional Discussion of Ablations and Training Overhead

The mask-ratio experiments characterize performance as the update budget changes. Component comparisons assess historical importance estimation and layer-adaptive allocation, and measurements of GPU memory and time per step quantify the additional training cost.

As shown in Table 3, the complete MAGM method achieves better overall performance than MAGM-IGI. MAGM-IGI uses current gradient magnitudes in both the layer aggregates and local rankings. This comparison assesses the contribution of historical smoothing.

The complete method also outperforms MAGM-UA. Although MAGM-UA retains historical importance estimates, it selects parameters globally rather than explicitly allocating budgets according to layer importance. Its lower performance suggests that parameter-level importance alone may distribute the update budget unevenly across layers. Explicitly modeling inter-layer differences can direct limited updates toward layers with greater contributions and improve parameter selection.

Overall, MAGM achieves a higher average score than either ablated variant. These results support the complementary roles of historical importance estimation and layer-adaptive allocation. The comparison with GMT further supports the effectiveness of the complete MAGM framework for sparse parameter updates.

Maintaining parameter-level EMA states and computing layer-wise mask ratios account for MAGM's additional overhead. The method introduces no new model components or trainable parameters, but the importance states require extra memory. The measured increase in time per step is comparatively small. At a fixed update ratio, this additional storage supports importance estimation across training steps and accompanies the reported performance gains.