DGMM: DYNAMIC GRADIENT MODELING WITH QUALITY-GUIDED MASKING FOR ADAPTIVE LLM FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient masking reduces redundant parameter updates during large language model (LLM) fine-tuning, but fixed global retention policies overlook variation across layers and training stages.We introduce DGMM, a dynamic gradient modeling algorithm with quality-guided masking that adaptively allocates gradient retention budgets across layers.DGMM integrates three complementary signals—directional sign bias, temporal stability, and inter-layer correlation—into a layer-wise gradient quality score. These scores are smoothed with an exponential moving average and used to allocate retention budgets within stage-aware bounds, while gradient magnitudes determine which elements are retained within each layer. Operating exclusively on optimizer-side gradients, DGMM preserves the model architecture, forward computation, training objective, and underlying optimizer. Experiments on task-specific and general-domain benchmarks show improvements of up to 30.8% over the strongest baseline on individual benchmarks. Controlled sweeps evaluate masking rates of up to 90% and reveal no change in profiler-reported training FLOPs. These findings support gradient-quality-guided budget allocation as an effective mechanism for adaptive gradient masking and underscore the distinction between sparse parameter updates and reduced training computation.

1 INTRODUCTION

Fine-tuning large language models (LLMs) commonly relies on dense parameter updates, which require substantial computational and memory resources. Recent efficient fine-tuning methods therefore investigate how to eliminate redundant updates while preserving adaptation capability. Gradient-based parameter selection methods, such as GMT, identify less important updates through gradient information and demonstrate that updating all parameters at every step is not always necessary (Li et al., 2025a). Adaptive parameter selection methods such as AdaDHP further explore fine-grained update allocation during fine-tuning (Liu et al., 2025a). Meanwhile, selective-update approaches show that partial model updates can maintain competitive performance while reducing training overhead (Hui et al., 2025). However, these approaches mainly focus on which parameters should be updated, while how to allocate update budgets across different layers and training stages remains insufficiently explored.

A common strategy for selective updating is to rank gradient coordinates based on their magnitudes. GMT retains large-magnitude gradient entries as important updates, while other gradient-aware methods, such as gradient grouping and salience-aware sparse fine-tuning, further exploit gradient-derived statistics to identify informative parameters (Li et al., 2025a;b; Liu et al., 2025b). Although coordinate-level ranking is effective within individual tensors, it does not characterize the optimization state of an entire layer. Different layers may exhibit distinct directional behaviors, temporal variations, and inter-layer dependencies even when they contain similar high-magnitude gradients. Therefore, magnitude alone is insufficient for determining how much update budget each layer should receive.

Recent layer-adaptive methods provide evidence that different components of LLMs may require different adaptation capacities. For example, Flexora and layer-wise analyses of supervised fine-tuning investigate non-uniform adaptation across layers (Wei et al., 2025; Zhao et al., 2026). However,

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.

these methods mainly adjust adaptation capacity or parameter allocation, rather than dynamically modeling the quality of gradients during optimization. Furthermore, update requirements change throughout training. Adaptive selection strategies in AdaDHP and salience-aware sparse fine-tuning reveal the non-stationary nature of parameter importance (Liu et al., 2025a;b). Excessive filtering at early stages may remove gradients required for initial adaptation, whereas maintaining dense updates in later stages may preserve unnecessary redundancy. An effective update strategy should therefore consider both the current layer state and the training stage.

To address these challenges, we propose DGMM, a dynamic gradient modeling framework that determines both how much gradient information should be retained and which gradient coordinates should be updated. DGMM summarizes each eligible layer using three complementary signals: directional sign bias, temporal stability, and inter-layer correlation. These statistics capture the dominant update direction, short-term gradient variation, and the relationship between a layer and the remaining network. The fused quality score is normalized and mapped to a bounded retention ratio, providing a layer-specific update budget.

This separation enables DGMM to handle cases where a layer contains many small but informative gradients or a few large but unstable gradients. By decoupling layer-level allocation from coordinate-level selection, DGMM preserves useful adaptation signals without introducing additional parameters or modifying the forward computation.

The main contributions are:

- **Dynamic layer-wise gradient quality modeling.** DGMM combines directional sign bias, temporal stability, and inter-layer correlation to characterize layer optimization states beyond instantaneous gradient magnitudes.
- **Quality-guided layer-adaptive retention.** The proposed quality score determines a bounded retention budget for each layer, while magnitude ranking is only applied to select coordinates within the allocated budget.
- **Stage-aware progressive optimization.** EMA smoothing and stage-dependent bounds enable a gradual transition from dense updates to stronger gradient filtering, while preserving the original optimizer and training procedure.

2 Related Work

Efficient LLM fine-tuning has followed three broad routes: restricting the adaptation space, updating selected parameters or layers, and modifying the gradient or optimizer state. DGMM falls in the last group, but its allocation uses layer information that a single gradient magnitude does not contain.

Low-rank and representation-efficient adaptation. LoRA freezes the pretrained weights and learns a low-rank update (Hayou et al., 2024). Later variants separate update magnitude from direction (Liu et al., 2024), or initialize the factors from principal singular components (Meng et al., 2024). Flexora adds critical-layer selection (Wei et al., 2025), AdaDHP chooses parameters at a finer granularity (Liu et al., 2025a), and Learnable Rank LoRA learns layer-specific adapter ranks instead of using a fixed rank allocation (Garg et al., 2026). ReFT intervenes in hidden representations while leaving the model weights fixed (Wu et al., 2024). In these approaches, the available update capacity is set mainly by the parameterization, rank, selected layers, or intervention location. They do not directly describe how the quality of a full-rank gradient changes across layers during training.

Selective parameter and layer updating. S2FT updates structured sparse subsets of heads and channels (Yang et al., 2024); AdaDHP selects important parameters (Liu et al., 2025a); LISA samples layers according to their importance (Pan et al., 2024); and HFT updates only part of the model (Hui et al., 2025). Flexora makes a related layer choice within low-rank adaptation (Wei et al., 2025). Together, these results show that dense updates are not always needed. Their selection, however, is usually made at the parameter or layer level and does not explicitly track how gradient quality evolves from one optimization step to the next.

Gradient- and optimizer-space fine-tuning. BAdam uses block coordinate descent to reduce memory use (Luo et al., 2024); LOMO combines gradient computation with the parameter update

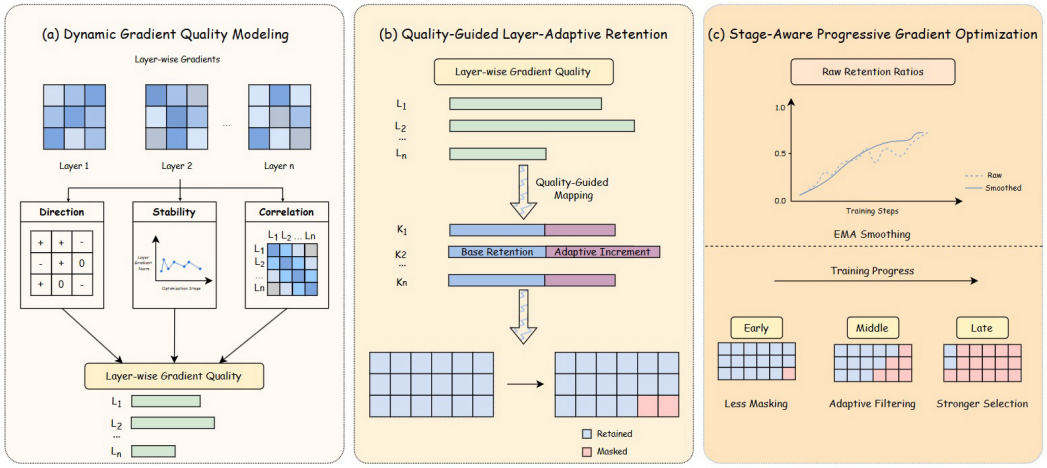


Figure 1: Overview of DGMM: quality allocates the layer budget, and magnitude selects coordinates within that budget.

(Lv et al., 2024); and GaLore projects gradients into a low-rank subspace (Zhao et al., 2024). SGG groups layer-wise gradient statistics for adaptive scaling (Li et al., 2025b), whereas GMT removes small-magnitude entries under a prescribed global ratio (Li et al., 2025a). These methods make the update signal itself part of the efficiency design, which is the setting most closely related to DGMM.

Limitations and motivation. These lines of work leave a more specific question open: how should a full-rank gradient budget be distributed across layers as training proceeds? Low-rank methods restrict the adaptation space, selective methods generally use a coarse activation rule, and magnitude masking often applies one global budget without a temporal or inter-layer description. The need for a changing layer treatment is also suggested by layer-wise SFT results showing depth-dependent sensitivity (Zhao et al., 2026) and by Evolving Parameter Isolation, which reports that parameter importance can drift over training (Lin et al., 2026).

3 Methodology

3.1 DGMM Algorithm

At the optimizer boundary, DGMM assigns a separate budget to every eligible transformer layer, as illustrated in Figure 1. The overall workflow consists of layer-wise quality estimation, adaptive retention budget allocation, and magnitude-based gradient selection. Multi-dimensional tensors with an unambiguous layer assignment are grouped together; one-dimensional tensors, embeddings, output heads, normalization parameters, and biases remain dense. Quality is estimated before coordinates are ranked by magnitude. The model, forward pass, loss, learning-rate schedule, and base optimizer are not altered.

The DGMM is designed around the optimizer step rather than the micro-batch boundary. When gradient accumulation is used, DGMM observes the accumulated gradient that would otherwise be passed directly to the base optimizer. This placement keeps its decision aligned with the actual parameter update and avoids changing the semantics of the loss or the accumulation schedule. Parameters that are not assigned to an eligible transformer layer follow the ordinary dense update path, so the dynamic policy is restricted to the tensors for which a layer-wise budget has a clear interpretation.

3.2 Dynamic Gradient Quality Modeling

DGMM characterizes the optimization state of each eligible transformer layer using three lightweight statistics. Throughout this section, l indexes an eligible transformer layer and t denotes the current

optimizer step. Rather than treating gradient magnitude as a complete quality signal, DGMM models sign structure, short-term temporal variation, and the relation of the current layer state to the rest of the network.

Directional sign bias. For layer l at step t, $P_l^{+,(t)}$ and $P_l^{-,(t)}$ denote the proportions of strictly positive and strictly negative gradient elements, respectively. The directional sign-bias score $D_l^{(t)}$ is their signed difference:

$$D_l^{(t)} = P_l^{+,(t)} - P_l^{-,(t)}, \qquad D_l^{(t)} \in [-1, 1]. \tag{1}$$

Because $D_l^{(t)}$ lies in $[-1, 1]$, a larger value indicates stronger positive sign asymmetry and contributes more positively to the subsequent layer-quality score.

Temporal stability. Let $G_l^{(t)}$ denote the aggregate ℓ_2 norm of the eligible gradients in layer l at step t, and let $H_l^{(t)}$ contain at most the 20 most recent values of this norm. The coefficient of variation $CV_l^{(t)}$ is computed from this history, and the bounded stability score is

$$S_l^{(t)} = 1 - \min\left(CV_l^{(t)}, 1\right), \qquad S_l^{(t)} \in [0, 1]. \tag{2}$$

Accordingly, a smaller $CV_l^{(t)}$ produces a larger $S_l^{(t)}$, so layers with more stable recent gradient norms receive higher stability scores.

Global inter-layer relation. For each layer, $P_l^{0,(t)}$ denotes the zero-gradient proportion, and the layer descriptor $f_l^{(t)}$ is formed from the positive, negative, and zero-sign ratios together with $S_l^{(t)}$. Stacking and feature-wise standardizing these descriptors across the L eligible layers yields $Z^{(t)}$. With descriptor dimension $d_f = 4$, the scaled relation matrix $R^{(t)}$ contains pairwise entries $R_{ab}^{(t)}$ between layers a and b. DGMM then summarizes the mean absolute off-diagonal relation as the shared global context $C^{(t)}$:

$$R^{(t)} = \frac{Z^{(t)} Z^{(t)\top}}{d_f}, \tag{3}$$

$$C^{(t)} = \mathrm{clip}_{[0,1]}\left(\frac{1}{L(L-1)} \sum_{a \neq b} \left|R_{ab}^{(t)}\right|\right). \tag{4}$$

At step t, the scalar $C^{(t)}$ is common to all eligible layers. The layer-to-layer differences in the final score consequently come mainly from $D_l^{(t)}$ and $S_l^{(t)}$.

The three signals are finally fused into the raw layer-quality score $Q_l^{(t)}$, which is then mapped to the normalized quality $q_l^{(t)} \in [0, 1]$:

$$Q_l^{(t)} = 3D_l^{(t)} + 2S_l^{(t)} + 2C^{(t)}, \tag{5}$$

$$q_l^{(t)} = \mathrm{clip}_{[0,1]}\left(\frac{Q_l^{(t)} + 3}{10}\right). \tag{6}$$

The normalized score $q_l^{(t)}$ controls how much gradient information layer l is allowed to retain; element-wise magnitude is used only afterward to determine which coordinates satisfy that allocated budget.

The terms have distinct roles: $D_l^{(t)}$ records signed balance, $S_l^{(t)}$ tracks recent scale variation, and $C^{(t)}$ provides a shared network reference. None is a substitute for magnitude. Together they set how much a layer may retain; magnitude subsequently selects which entries occupy that budget. We use fixed weights of 3:2:2 and clip the affine normalization, preventing a single atypical statistic from producing an out-of-range target.

Stage-aware retention schedule used by the default configuration.

Optimizer steps	$K_{\min}^{(t)}$	$K_{\max}^{(t)}$	Operation
$t \leq 500$	–	–	Masking bypassed (dense warm-up)
$501 \leq t \leq 999$	0.95	0.99	Conservative masking
$t \geq 1000$	0.89	0.99	Full quality-guided range

3.3 Quality-Guided Retention and Masking

We first turn the normalized quality $q_l^{(t)}$ into a raw retention target $K_{l,\mathrm{raw}}^{(t)}$. The target is smoothed with an exponential moving average controlled by α (set to 0.99 by default), and the resulting value is clipped to the bounds for the current training stage. Thus $K_{l,\mathrm{EMA}}^{(t)}$ is an intermediate state, while $K_l^{(t)}$ is the value actually used for masking:

$$K_{l,\mathrm{raw}}^{(t)} = 0.89 + 0.10 q_l^{(t)}, \tag{7}$$

$$K_{l,\mathrm{EMA}}^{(t)} = \alpha K_l^{(t-1)} + (1-\alpha) K_{l,\mathrm{raw}}^{(t)}, \tag{8}$$

$$K_l^{(t)} = \mathrm{clip}\Big(K_{l,\mathrm{EMA}}^{(t)}, K_{\min}^{(t)}, K_{\max}^{(t)}\Big). \tag{9}$$

The term $K_l^{(t-1)}$ is the final, stage-constrained value stored at the preceding optimizer step. It is this value, rather than an unconstrained raw target, that enters the next EMA update.

The default run is dense through step 500. Retention is restricted to $[0.95, 0.99]$ during steps 501–999 and to $[0.89, 0.99]$ from step 1000 onward. These are conservative, reproducible bounds rather than claimed optima. The upper limit 0.99 matches the maximum of $K_{l,\mathrm{raw}}^{(t)} = 0.89 + 0.10 q_l^{(t)}$ and prevents a masking-active layer from reverting to a fully dense update; the lower limit caps default removal at 11%.

Once masking is active, $\tau_l^{(t)}$ denotes the magnitude threshold estimated for layer l from a bounded sample of at most 100,000 absolute gradient values. For an eligible parameter tensor i belonging to the layer index set $\mathcal{I}_l$, $g_i^{(t)}$ denotes its gradient tensor and $M_i^{(t)}$ denotes the corresponding binary mask:

$$M_i^{(t)} = \mathbf{1}\Big\{\Big|g_i^{(t)}\Big| \geq \tau_l^{(t)}\Big\}, \qquad i \in \mathcal{I}_l. \tag{10}$$

The indicator in Equation 10 is applied element-wise and returns one for entries whose absolute magnitude is at least the layer threshold, and zero otherwise. The implementation optionally supports soft attenuation, where s denotes the scaling factor applied to below-threshold entries, together with a gradual activation ramp. DGMM operates on accumulated gradients at optimizer-step time and then invokes the original base optimizer.

Warm-up fills the recent-norm histories before masking affects an update. The narrow middle range limits the initial change, and the full preset range begins at step 1000. EMA handles step-to-step variation, clipping enforces the stage bounds, and bounded threshold samples keep selection manageable for large tensors. Every eligible tensor assigned to a layer uses the same layer budget.

Algorithm 1 records the update order. Statistics are collected from the common pre-mask state; the shared relation and all layer budgets are then computed before any threshold is applied. The unchanged optimizer runs last. Hence, a mask for one layer cannot affect another layer's score within the same step.

4 Experiments

4.1 Experimental Setup

We test DGMM in two settings. The task-specific experiments use MISTRAL-7B and DEEPSEEK-CODER-BASE-6.7B on HumanEval, HumanEval+, MBPP, MBPP+, GSM8K, and MATH. The broader evaluation uses LLAMA2-7B after supervised fine-tuning on TULU V2 and after preference

Algorithm 1 DGMM

Require: $\theta^{(0)}$, accumulated gradients $G^{(t)}$, layer groups $\{\mathcal{I}_l\}$, EMA coefficient α, stage bounds, and base optimizer $\mathcal{O}$

Ensure: Updated parameters $\theta^{(T)}$

Set the layer histories and retention states to their initial values.
for optimizer steps $t = 0, \ldots, T-1$ **do**
  Copy the accumulated gradients and group eligible tensors by transformer layer.
  for each layer l **do**
    $D_l^{(t)} \leftarrow P_l^{+,(t)} - P_l^{-,(t)}$, $D_l^{(t)} \in [-1, 1]$
    $S_l^{(t)} \leftarrow 1 - \min(CV_l^{(t)}, 1)$, $S_l^{(t)} \in [0, 1]$
  end for
  $C^{(t)} \leftarrow \mathrm{clip}_{[0,1]}\left(\frac{1}{L(L-1)} \sum_{a \neq b} |R_{ab}^{(t)}|\right)$
  for each layer l **do**
    $Q_l^{(t)} \leftarrow 3D_l^{(t)} + 2S_l^{(t)} + 2C^{(t)}$
    $q_l^{(t)} \leftarrow \mathrm{clip}_{[0,1]}\left(\frac{Q_l^{(t)}+3}{10}\right)$
    $K_{l,\mathrm{raw}}^{(t)} \leftarrow 0.89 + 0.10 q_l^{(t)}$
    $K_{l,\mathrm{EMA}}^{(t)} \leftarrow \alpha K_l^{(t-1)} + (1-\alpha) K_{l,\mathrm{raw}}^{(t)}$
    $K_l^{(t)} \leftarrow \mathrm{clip}(K_{l,\mathrm{EMA}}^{(t)}, K_{\min}^{(t)}, K_{\max}^{(t)})$
    if masking is active **then**
      Estimate $\tau_l^{(t)}$ and apply $M_i^{(t)} = \mathbf{1}\{|g_i^{(t)}| \geq \tau_l^{(t)}\}$ for $i \in \mathcal{I}_l$.
    end if
  end for
  Update parameters with the original base optimizer.
end for
return the updated model.

optimization on UltraFeedback, with MMLU, GSM8K, BBH, TyDiQA, TruthfulQA, and HumanEval as the reported benchmarks. The comparison algorithms include the pretrained model where applicable, dense SFT, HFT, GMT, and DGMM. HFT limits the trainable subset. The task-specific group concentrates optimization on a narrow capability; the general-domain group checks whether the update rule preserves broader behavior.

4.2 Baselines

We compare DGMM with representative baselines covering full-parameter fine-tuning, parameter-efficient updating, and selective-update optimization. SFT(Zhao et al., 2026) updates all model parameters and serves as the full-update reference. HFT(Hui et al., 2025) performs partial parameter updating by restricting the trainable subset, reducing unnecessary parameter changes while maintaining adaptation capability. GMT(Li et al., 2025a) adopts gradient-based parameter selection and removes low-importance gradient entries under a predefined selection ratio. DGMM is compared against these methods to evaluate whether quality-guided gradient retention can provide a more adaptive update strategy across layers and training stages. For general-domain experiments, pretrained models are also reported as the initialization reference.

4.3 Main Results

4.3.1 Task-Specific Domains

Table 1 reports task-specific performance and shows that DGMM's advantage depends on the backbone and benchmark rather than being uniform across tasks.

For MISTRAL-7B code generation, GMT has the largest code average at 41.05%. DGMM is close at 40.70%, 2.70% above dense SFT, and gives the best HumanEval+ result (32.90%). The picture is

Table 1: Task-specific performance on code generation and mathematical reasoning. The code average is computed over HumanEval, HumanEval+, MBPP, and MBPP+, while the math average is computed over GSM8K and MATH.

Backbone	Method	HumanEval	HumanEval+	MBPP	MBPP+	Code Average	GSM8K	MATH	Math Average
MISTRAL-7B	Pre-trained	30.3	25.9	46.0	42.0	36.05	–	–	–
	SFT	32.3	26.2	55.1	44.9	39.625	16.0	19.2	17.6
	GMT	39.6	31.1	51.9	41.6	41.05	23.6	17.8	20.7
	DGMM	38.4	32.9	50.6	40.9	**40.7**	23.0	23.2	**23.1**
DEEPSEEK-CODER BASE-6.7B	Pre-trained	42.2	35.2	60.9	51.2	47.38	3.0	10.5	6.75
	SFT	47.0	39.6	64.4	53.4	51.1	4.6	11.4	8.0
	GMT	44.5	38.4	64.9	54.6	50.6	4.8	12.0	8.4
	DGMM	60.4	51.8	63.2	53.6	**57.25**	4.6	12.2	**8.4**

Table 2: General-domain performance of LLAMA2-7B under supervised fine-tuning on TULU V2 and the subsequent preference-optimization setting on UltraFeedback. Average denotes the mean over the six reported benchmarks.

Setting	Model	Method	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	Average
Pre-trained	LLAMA2-7B	Pre-trained	32.00	13.50	39.62	51.87	40.33	22.10	33.24
SFT on TULU V2	LLAMA2-7B	SFT	39.12	17.59	40.56	52.69	42.63	17.10	34.95
		HFT	36.58	16.67	39.24	52.00	41.78	18.06	34.06
		GMT	39.25	17.21	40.65	53.04	42.42	16.50	**34.85**
		DGMM	38.88	17.44	40.44	52.32	42.50	17.10	34.78
DPO on UltraFeedback	LLAMA2-7B	SFT	39.13	17.97	40.58	52.55	42.63	15.20	34.68
		HFT	39.56	17.20	37.85	51.28	41.78	17.77	34.24
		GMT	39.23	17.74	40.58	53.02	42.32	15.20	34.68
		DGMM	39.23	17.29	39.99	52.26	42.51	17.70	**34.83**

different for DEEPSEEK-CODER-BASE-6.7B: DGMM reaches 57.25%, compared with 51.10% for SFT and 50.60% for GMT. Most of this gap comes from HumanEval and HumanEval+, where DGMM scores 60.40% and 51.80%.

On the mathematical benchmarks, DGMM reaches a 23.10% average with MISTRAL-7B, versus 17.60% for SFT and 20.70% for GMT. Its GSM8K and MATH scores are 23.00% and 23.20%. With DEEPSEEK-CODER-BASE-6.7B, DGMM and GMT both average 8.40%; DGMM is higher on MATH (12.20%), while GMT remains higher on GSM8K (4.80%).

The backbone matters. With MISTRAL-7B, DGMM is near the best code average, leads on HumanEval+, and gives the strongest mathematical average. With DEEPSEEK-CODER-BASE-6.7B, the clearest gains occur on HumanEval and HumanEval+, while MBPP and MBPP+ remain close to the magnitude baselines. The result is therefore a redistribution of gains across backbone–benchmark pairs, not a uniform rise in every column.

4.3.2 General Domain

Table 2 evaluates the broader general-domain setting, where SFT, GMT, and DGMM are much closer than in the task-specific experiments.

With supervised fine-tuning on TULU V2, dense SFT gives the highest average (34.95%). DGMM is at 34.78%, about 0.50% below SFT and 0.20% below GMT, and remains above HFT by roughly 2.10%. The methods are close overall; DGMM preserves the broad-domain level but is not the leading method in every column.

After preference optimization on UltraFeedback, DGMM has the highest average at 34.83%, compared with 34.68% for both SFT and GMT and 34.24% for HFT. Its HumanEval score is 17.70%, whereas SFT and GMT are at 15.20%. The improvement is modest in this setting, and the more pronounced gains still occur in the task-specific experiments.

The general-domain averages occupy a narrow range, although the leading method changes. DGMM is 0.17 below dense SFT after supervised fine-tuning and has the best average after preference optimization. Its small gains and losses are spread across knowledge, reasoning, multilingual QA, truthfulness, and code generation, so no single column determines the comparison.

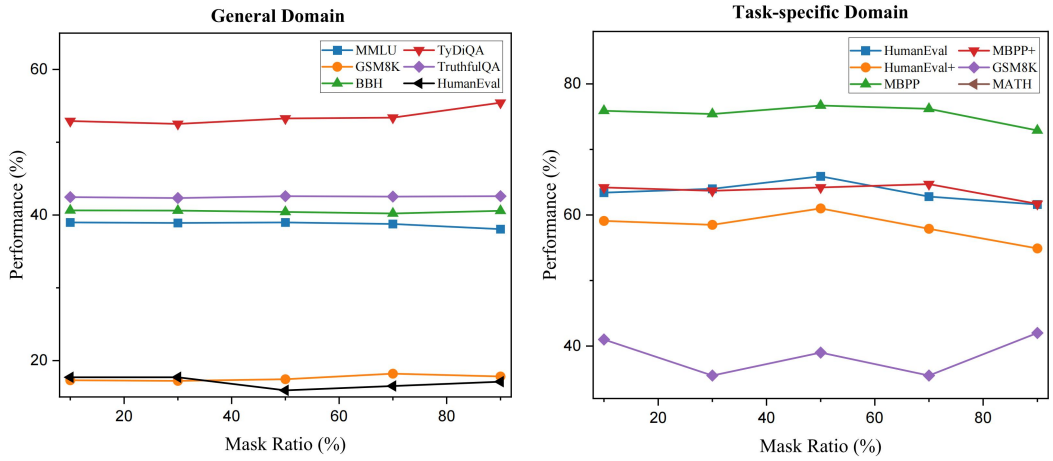


Figure 2: Sensitivity of DGMM to the external mask ratio: general-domain results are shown on the left and task-specific results on the right.

Table 3: Efficiency measurements over 100 optimizer steps. Speed is reported in samples/s, FLOPs are the profiler-reported model computation in units of 10^{18} operations, and runtime is wall-clock time in seconds.

Model	Domain	Method	Training Speed (samples/s)	FLOPs (10^{18})	Runtime (s)
MISTRAL-7B	Code Generation	SFT	2.783	0.140735	**1149.922**
		GMT	2.621	0.140735	1220.828
		DGMM	**2.592**	0.140735	1234.787
LLAMA2-7B	General Domain	SFT	2.735	0.027283	**292.532**
		GMT	**2.228**	0.027283	359.131
		DGMM	2.467	0.027283	324.312

4.4 Analysis of Mask Ratio

We vary the external mask-ratio budget from 10% to 90% to test sensitivity beyond a single sparsity point.

General- and task-specific sensitivity. Figure 2 illustrates the sensitivity of different benchmarks to the masking ratio under general-domain and task-specific evaluations. In the general-domain curves, MMLU remains around 38%–39%, BBH around 40%, and TruthfulQA around 42%; TyDiQA is somewhat higher at the larger mask ratios. There is no systematic collapse across the 10%–90% sweep. The task-specific curves move more: several code metrics are strongest near 50%, HumanEval, HumanEval+, and MBPP fall at 90%, and the mathematics results are non-monotonic. The preferred sparsity level therefore depends on the task rather than being a single global constant.

The sweep exposes the weakness of a single global ratio. A setting that works for the broad evaluation can be poor for code or mathematics, and neither 10% nor 90% is consistently preferred. General-domain curves are flatter, whereas task-specific curves depend more strongly on the backbone and benchmark. DGMM instead assigns budgets by layer within stage-dependent bounds.

4.5 Analysis of Efficiency

We compare dense SFT, GMT, and DGMM over 100 optimizer steps and report throughput, profiler-reported model FLOPs, and wall-clock runtime for MISTRAL-7B code generation and LLAMA2-7B general-domain fine-tuning in Table 3.

Table 4: DGMM ablation by component. The average is the unweighted mean of the five benchmarks shown.

Variant	TyDiQA	MMLU	TruthfulQA-MC2	HumanEval	HumanEval+	Average
DGMM	52.45	41.40	43.86	18.10	15.70	**34.30**
w/o Direction	52.33	41.02	43.80	16.50	14.60	33.65
w/o Stability	52.32	41.27	43.82	17.70	15.20	34.06
w/o Correlation	52.31	41.17	43.77	17.10	15.20	33.91

The profiler reports the same model FLOPs for the three methods in each setting. For MISTRAL-7B, DGMM processes 2.592 samples/s, compared with 2.783 for SFT and 2.621 for GMT, and takes 1234.787 s. This is a 6.90% throughput reduction relative to SFT and a 1.10% reduction relative to GMT. For LLAMA2-7B, DGMM reaches 2.467 samples/s with a runtime of 324.312 s; relative to GMT, throughput is about 10.70% higher and runtime about 9.70% lower. The extra optimizer-side work therefore depends on the workload, whereas the reported model FLOPs do not change.

These figures describe update selection, not hardware acceleration. DGMM still runs the forward pass, backward pass, and dense base optimizer; it additionally maintains short histories, computes layer statistics, estimates bounded thresholds, and applies masks. Profiler-reported model FLOPs therefore remain unchanged. Relative to GMT, the implementation is slightly slower for MISTRAL-7B but faster for LLAMA2-7B, showing that the overhead follows the workload and execution path.

4.6 Ablation Study

Table 4 removes Direction (directional sign bias), Stability (temporal stability), or Correlation (global inter-layer relation) from DGMM and reports the unweighted mean over five benchmarks.

The full configuration is the best row on all five benchmarks and has an average of 34.30%. Removing Direction lowers the average to 33.65% (1.90% relative); HumanEval and HumanEval+ fall from 18.10% and 15.70% to 16.50% and 14.60%. Without Stability or Correlation, the averages become 34.06% and 33.91%, corresponding to relative decreases of 0.70% and 1.14%. Each reduced version is below the full model in every displayed column.

Direction has the largest individual effect, yet Stability and Correlation add information absent from a one-step sign count. The former reflects recent scale changes; the latter relates the current layer descriptors to one another. Every reduced variant is lower in every displayed column, not only in the average. Under the reported configuration, the fused estimate is therefore preferable to using one statistic alone.

5 Conclusion

DGMM provides a algorithm for adaptive gradient masking that separates layer-wise retention budget allocation from coordinate selection at the optimizer interface. Directional sign bias, temporal stability, and inter-layer correlation jointly determine the retention budget for each eligible layer, while gradient magnitudes determine which coordinates are retained within that budget. Exponential moving average smoothing and stage-aware bounds stabilize the allocation process while preserving the model architecture, forward computation, training objective, learning-rate schedule, and underlying optimizer.

DGMM yields its largest gains on task-specific benchmarks, while general-domain performance remains comparable to dense supervised fine-tuning (SFT) and GMT. Sensitivity to the masking ratio is limited in the general domain but varies more substantially across task-specific benchmarks. Profiler-reported training FLOPs remain unchanged, and optimizer-side costs depend on the workload, underscoring the distinction between update sparsity and computational savings. Future work could explore learning these settings adaptively and combining quality-guided allocation with structured, hardware-aware masking to translate sparse updates into practical computational savings.

AI USE STATEMENT

Generative AI tools were used solely for language polishing after the manuscript had been completed by the authors.

REFERENCES

H. Liu, C. Li, X. Zhang, F. Zhang, F. Ma, W. Wang, and H. Yu. AdaDHP: Fine-grained fine-tuning via dual hadamard product and adaptive parameter selection. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9492–9504, 2025a.

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, 2025a. No. 23.

T. Hui, Z. Zhang, S. Wang, W. Xu, Y. Sun, and H. 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.

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, 2025b.

X. Yang, J. Leng, G. Guo, J. Zhao, R. Nakada, L. Zhang, H. Yao, and B. Chen. S2FT: Efficient, scalable and generalizable LLM fine-tuning by structured sparsity. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

C. Wei, Y. Shu, Y. T. He, and F. Yu. Flexora: Flexible low-rank adaptation for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14643–14682, 2025.

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*, volume 235, pp. 32100–32121, 2024.

S. Hayou, N. Ghosh, and B. Yu. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 17783–17806, 2024.

F. Meng, Z. Wang, and M. Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Z. Wu, A. Arora, Z. Wang, A. Geiger, D. Jurafsky, C. D. Manning, and C. Potts. ReFT: Representation finetuning for language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

R. Pan, X. Liu, S. Diao, R. Pi, J. Zhang, C. Han, and T. Zhang. LISA: Layerwise importance sampling for memory-efficient large language model fine-tuning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Q. Luo, H. Yu, and X. Li. BAdam: A memory efficient full parameter optimization method for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

K. Lv, Y. Yang, T. Liu, Q. Guo, and X. Qiu. Full parameter fine-tuning for large language models with limited resources. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8187–8198, 2024.

J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 61121–61143, 2024.

S. Li, J. Tian, Z. Wang, X. Jin, Z. Liu, W. Zhang, and D. Xu. Taming LLMs with gradient grouping. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3260–3279, 2025b.

Q. Zhao, X. Gong, X. Chen, Z. Kang, and X. Li. A layer-wise analysis of supervised fine-tuning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9981–9992, 2026.

Z. Lin, C. Xue, D. Liang, X. Han, P. Liu, X. Wu, L. Jiang, Y. Lu, B. Simons, S. Liang, and M. Peng. Parameter importance is not static: Evolving parameter isolation for supervised fine-tuning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32846–32864, 2026.

A. Garg, S. Lucey, and H. Saratchandran. Parameter-efficient fine-tuning with learnable rank. arXiv preprint arXiv:2606.04325, 2026.