GDPLoRA: Partitioned Low-Rank Adaptation with Gap-Driven Local Rank Allocation

Anonymous authors
Paper under double-blind review

Abstract

Low-Rank Adaptation (LoRA) applies a uniform rank to each weight projection, implicitly assuming that all parameters of a projection require equal adaptation capacity. We show that this assumption does not hold at a finer granularity: local submatrices within a single projection exhibit markedly different spectral concentration, so a monolithic low-rank factorization fits them unevenly. Crucially, this structural signal is already present in the frozen pretrained weights, before any fine-tuning. Building on this observation, we propose **Gap-Driven Partitioned LoRA (GDPLoRA)**, a fine-grained adaptation framework that tailors rank allocation to local weight sub-structures. GDPLoRA partitions each frozen weight matrix into functionally cohesive domains via direction-based clustering, then redistributes a fixed per-projection rank budget according to local spectral concentration. Because domains and rank assignments are determined offline from the frozen weights, GDPLoRA requires no input-dependent routing and preserves exact mergeability into the pretrained weights, leaving inference overhead unchanged. Experiments demonstrate that GDPLoRA outperforms competitive PEFT baselines across multiple backbones and tasks.

1 Introduction

While foundation models have achieved remarkable success, adapting them to downstream tasks via full-parameter fine-tuning incurs prohibitive computational and memory costs. Parameter-efficient fine-tuning (PEFT) addresses this bottleneck by updating only a small subset of parameters while keeping the vast majority of pretrained weights frozen (Houlsby et al., 2019; Lester et al., 2021; Li & Liang, 2021). Among various PEFT strategies, Low-Rank Adaptation (LoRA) (Hu et al., 2022) has emerged as the dominant choice by keeping the pretrained weights frozen and injecting trainable low-rank matrices to approximate the task-specific weight update.

Given its simplicity and effectiveness, LoRA has inspired a growing body of work that can be broadly grouped into three directions. The first adjusts rank allocation or initialization at the whole-matrix level, as in AdaLoRA (Zhang et al., 2023a), PiSSA (Meng et al., 2024), and DoRA (Liu et al., 2024b). The second introduces finer-grained adaptation blocks within a matrix, as in GraLoRA (Jung et al., 2025). The third combines LoRA with mixture-of-experts (MoE) routing (Zadouri et al., 2024; Liu et al., 2024a; Dou et al., 2024; Li et al., 2024; Wu et al., 2024), using input-dependent gating to select among multiple low-rank experts. Despite their respective merits, these methods share a common limitation in that each adaptation unit is still assigned a uniform rank without accounting for the local spectral structure within a single projection. This raises a fundamental question: *is a uniform rank assignment truly optimal, or do distinct local regions within a single projection inherently demand tailored adaptation capacities?*

To answer this question, we examine the empirical weight updates produced by full-parameter fine-tuning. Our analysis reveals that local channel groups within a single projection possess markedly different spectral profiles. Specifically, submatrices whose updates spread energy across many singular directions suffer disproportionately large reconstruction errors under a shared, matrix-wide low-rank approximation. This confirms that a single monolithic factorization cannot adequately capture localized spectral structures. However, full-parameter updates are available only after costly downstream training and therefore cannot guide adapter design in advance. The key challenge is thus to identify an analogous structural signal that is already present in the frozen pretrained weights.

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.

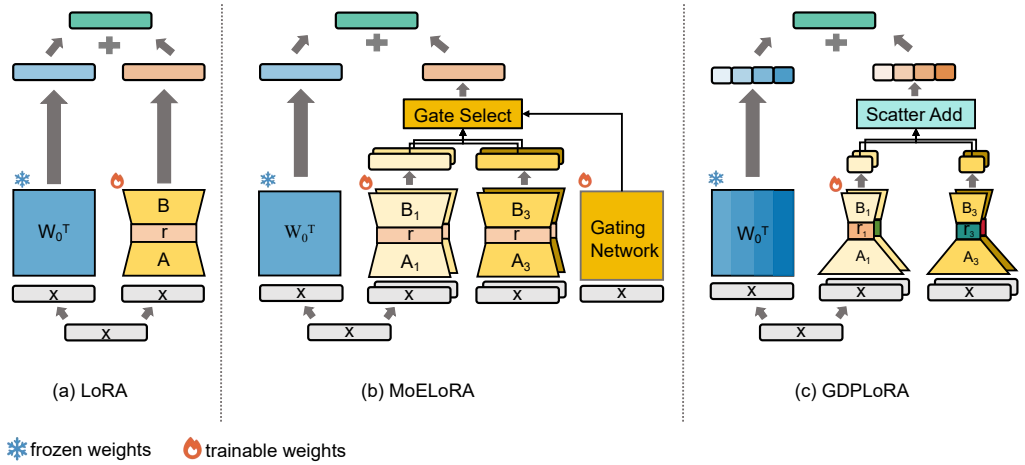


Figure 1: Comparison of different LoRA paradigms. (a) Standard LoRA applies a single low-rank factorization globally to the target layer. (b) MoE-based LoRA employs multiple experts with token-dependent routing, which incurs runtime overhead and prevents zero-overhead merging. (c) GDPLoRA partitions the weight matrix into functionally coherent domains, each with an independent LoRA branch. Branch ranks reflect local spectral concentration under a fixed budget. Colors denote domains; block heights indicate allocated ranks.

Motivated by prior findings that Transformer neurons exhibit directional clustering and functional modularity (Clark et al., 2019; Geva et al., 2021; Zhang et al., 2022), we group frozen weights by directional similarity and analyze their local spectra. The results confirm that these weight-derived domains exhibit consistent differences in spectral concentration *before* any fine-tuning takes place, providing a readily available, data-free cue for allocating capacity to each region.

Building on these insights, we propose **Gap-Driven Partitioned LoRA (GDPLoRA)**, a fine-grained adaptation framework that allocates capacity according to local sub-structures within each target projection. As illustrated in Figure 1, GDPLoRA first partitions frozen weights into disjoint, functionally cohesive domains via size-regularized directional clustering, then measures the spectral concentration of each domain, and finally redistributes a fixed per-projection rank budget accordingly. Because all domain boundaries and rank assignments are determined offline before fine-tuning, GDPLoRA requires no gating parameters, incurs no runtime dispatch overhead, and preserves exact weight mergeability for zero-overhead inference. Experiments on RoBERTa-large, LLaMA-3-8B, and Qwen3-8B across natural language understanding, mathematical reasoning, code generation, and multi-turn instruction following show that GDPLoRA improves aggregate performance over prior LoRA-based methods at comparable trainable-parameter budgets.

Our main contributions are as follows:

- We reveal widespread intra-layer spectral heterogeneity in both full-parameter fine-tuning updates and frozen base weights, showing that a monolithic low-rank parameterization yields uneven approximation quality across local submatrices.
- We propose GDPLoRA, a fine-grained adaptation framework that combines directional weight clustering with gap-driven rank redistribution and principal singular-direction initialization, enabling localized capacity allocation without any dynamic routing.
- Across three backbones and four task families, GDPLoRA improves aggregate performance over competitive PEFT baselines at comparable trainable-parameter budgets.

2 Related Work

Structure Inside Dense and MoE Architectures. A line of research studies how dense Transformers are organized internally, at a granularity finer than a single projection. Analyses of attention heads and feed-forward network (FFN) neurons have documented substantial functional specialization, redundancy, and structured activation patterns (Clark et al., 2019; Voita et al., 2019; Geva et al., 2021; 2022; Dalvi et al., 2020; Zhang et al., 2023b; Liu et al., 2023b; Dai et al., 2022). Related studies report persistent activation sparsity and neuron specialization that varies with the task or the language (Zhang

et al., 2024; Tan et al., 2024; Wang et al., 2022). A complementary direction exploits this structure to reorganize dense models into expert-based architectures: MoEfication (Zhang et al., 2022) groups FFN neurons by pretrained-weight similarity or activation co-occurrence, and LLaMA-MoE (Zhu et al., 2024) partitions pretrained FFN parameters into experts, followed by continual pretraining and router learning. These studies establish that useful substructure exists within dense pretrained layers. Their objectives, however, are model interpretation or the conversion of a dense model into a sparsely activated one, rather than the allocation of fine-tuning capacity inside a frozen projection.

Low-Rank Adaptation (LoRA). Most LoRA variants operate at the level of an entire target matrix (Valipour et al., 2023; Yang et al., 2024; Lingam et al., 2024). LoRA (Hu et al., 2022) attaches one low-rank update to each projection; AdaLoRA (Zhang et al., 2023a) reallocates a global rank budget across matrices during training; PiSSA (Meng et al., 2024) initializes the factors from the principal singular components of the pretrained weights; and DoRA (Liu et al., 2024b) decomposes the update into magnitude and direction. These methods differ in rank allocation, initialization, and parameterization, but the unit of adaptation remains the complete projection. GraLoRA (Jung et al., 2025) moves to a finer granularity by applying independent low-rank updates to local blocks and demonstrates the benefit of localizing adaptation; its blocks, however, follow a fixed geometric layout rather than structure found in the pretrained weights.

MoE-based LoRA. Another family of methods combines LoRA with mixture-of-experts mechanisms. MoLoRA, MoELoRA, and related approaches (Zadouri et al., 2024; Liu et al., 2024a; Dou et al., 2024; Li et al., 2024; Wu et al., 2024) construct several low-rank experts for the same projection and use input-dependent routing to select or combine them, so their diversity comes from routing among matrix-wide experts at the cost of extra router parameters and selection machinery. GDPLoRA takes a different route: it derives disjoint local domains from the frozen weights, redistributes a fixed rank budget across them, and keeps the structure static throughout fine-tuning, obtaining heterogeneous adaptation within a matrix without token-dependent expert routing.

3 MOTIVATION

Vanilla LoRA applies a global low-rank update to the entire target weight matrix, overlooking potential variations across its local regions. We first investigate whether such regional differences actually emerge during downstream training.

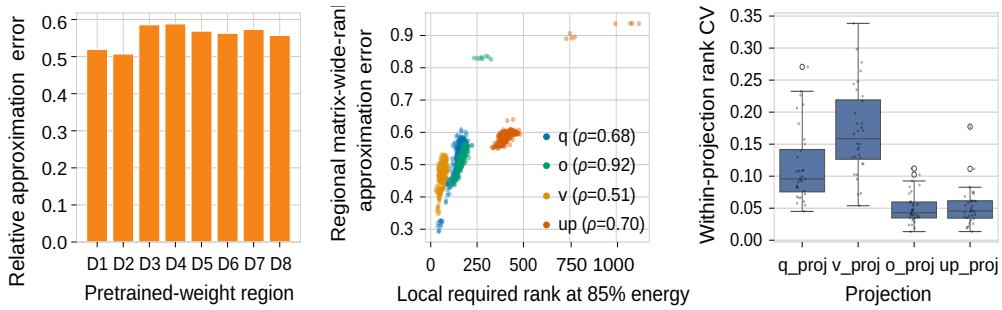
Let $\mathbf{W}_0$ denote a pretrained matrix and $\mathbf{W}_{\mathrm{FT}}$ its counterpart after full fine-tuning (Full FT). We define the corresponding full Full FT update as $\Delta\mathbf{W}_{\mathrm{FT}} = \mathbf{W}_{\mathrm{FT}} - \mathbf{W}_0$. For a local region S_i, we take the corresponding row submatrix $\Delta\mathbf{W}_i = \Delta\mathbf{W}_{\mathrm{FT}}[S_i, :]$. To compare local regions under a common criterion, we adopt an energy-retention threshold of 0.85. Let $\sigma_{i,j}$ denote the singular values of $\Delta\mathbf{W}_i$, and define

$$k_i^{(0.85)} = \min\left\{ k : \frac{\sum_{j=1}^{k} \sigma_{i,j}^2}{\sum_j \sigma_{i,j}^2} \geq 0.85 \right\}. \tag{1}$$

Here, $k_i^{(0.85)}$ is the number of singular directions required to retain 85% of the local Full FT update energy. A larger value indicates that the update energy is spread across more singular directions, whereas a smaller value indicates stronger spectral concentration. The quantity measures the compressibility of an *observed* Full FT update; it is not an estimate of the LoRA rank that is optimal for a downstream task.

The relevant question is not only whether local regions change differently during fine-tuning, but also whether a single matrix-wide low-rank representation fits their local spectra uniformly. To test this, we retain the leading $K_{\mathrm{svd}} = 128$ singular components of the complete Full FT update and measure the normalized Frobenius reconstruction error within each local region.

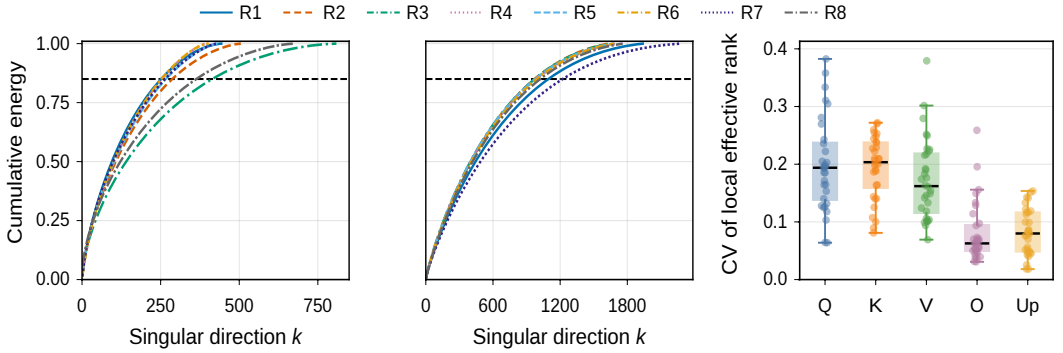
Figure 2(a) shows that local regions retain different amounts of error when the complete Full FT update is approximated by a single matrix-wide rank-K truncated SVD; Figure 2(b) shows that regions needing more singular directions to retain the same energy fraction incur larger error; and Figure 2(c) shows that this requirement varies substantially within a projection. A single rank-K truncated SVD of the complete update therefore leaves systematically different residuals across local regions, which motivates adaptation at a finer granularity. Because the Full FT update is available



(a) Matrix-wide rank-128 coveragec (b) Whole-model relationship (c) Full-FT local-rank heterogeneity

Figure 2: Local heterogeneity of Full FT updates. (a) Regional reconstruction error under a shared matrix-wide truncated-SVD approximation. (b) Relation between $k_i^{(0.85)}$ and the regional reconstruction error. (c) Distribution of the within-projection spread of $k_i^{(0.85)}$.

only after downstream training, it cannot be used to build the adaptation structure in advance; we instead ask whether an analogous signal can be read from the frozen pretrained weights.



(a) q_proj, Layer 19 (b) up_proj, Layer 9 (c) Across-layer heterogeneity

Figure 3: Local spectral heterogeneity in frozen pretrained weights. (a) Cumulative spectral-energy profiles of local regions in a representative attention projection. (b) Corresponding profiles in a representative FFN projection. (c) Distribution of the within-projection spread of the full-spectrum energy-retention rank across Transformer layers.

We therefore examine the full local spectra of frozen pretrained projections under the same energy-retention criterion ($\tau = 0.85$), computed by FP32 Gram-matrix eigendecomposition. In Figure 3(a), local submatrices of a representative attention projection follow clearly different cumulative energy curves—some concentrate most of their energy in a few leading directions, others spread it over many. Figure 3(b) shows the same behavior in FFN projections, and Figure 3(c) summarizes the within-projection spread across layers. Local spectral heterogeneity is thus a pervasive property of pretrained Transformers rather than an artifact of particular layers or projections.

The two analyses are complementary: the Full FT study motivates local parameterization, while the frozen-weight study supplies a signal available before training. GDPLoRA uses the latter as a task-independent prior for rank allocation. Its relevance to adaptation is further supported by the initialization of Section 4.3, where local pretrained singular directions determine how the task gradient first enters each adapter (Appendix A.5). That connection does not prescribe an allocation direction, so favoring domains with smaller descriptors remains a hypothesis, which we test in Section 5.4.

4 Methodology

Motivated by Section 3, GDPLoRA adapts over static local domains instead of treating a projection as a single adaptation unit. As summarized in Algorithm 1, it constructs G domains from the frozen matrix $\mathbf{W}_0$, characterizes their spectral structure, redistributes a fixed rank budget R across them, and attaches an independent low-rank branch to each domain. Domain boundaries and ranks are fixed before fine-tuning and held constant during training.

Algorithm 1 GDPLoRA Construction for a Target Projection

Require: Frozen weight matrix $\mathbf{W}_0$; number of domains G; rank budget R; spectral threshold τ; redistribution coefficient λ; clipping bound c; scaling ratio ρ
Ensure: Local domains and adapter parameters $\{S_i, r_i, \mathbf{A}_i, \mathbf{B}_i\}_{i=1}^{G}$

```
Partition target matrix into G domains {S_i}_{i=1}^G via size-regularized cosine clustering on output rows
For SwiGLU FFNs, apply the derived channel partition simultaneously to up_proj and gate_proj rows, and to down_proj columns
for i = 1, ..., G do
    Compute truncated spectral descriptor e_i via Eq. 2
end for
Compute mean descriptor ē = (1/G) Σ_i e_i and uniform reference rank r̄ = R/G
for i = 1, ..., G do
    Compute normalized spectral deviation γ_i and real-valued rank proposal r̃_i via Eq. 3
    Compute uncorrected integer rank r̂_i = max{1, round(r̃_i)}
end for
Perform residual budget correction on {r̂_i} to obtain final integer ranks {r_i} satisfying Σ_{i=1}^G r_i = R
for i = 1, ..., G do
    Initialize input factor A_i from leading local right singular vectors V_i[:, 1:r_i]^⊤
    Initialize output factor B_i = 0, and set LoRA scaling α_i = ρ r_i
end for
return {S_i, r_i, A_i, B_i}_{i=1}^G
```

4.1 Weight-Derived Local Adaptation Domains

We construct the adaptation units directly from the frozen weights. For attention projections, the rows of $\mathbf{W}_0$ are ℓ_2-normalized and grouped by size-regularized spherical K-Means: the cosine objective groups rows by direction rather than magnitude (Dhillon & Modha, 2001), and the size term penalizes assignment to already large groups so that a few domains do not absorb most coordinates. The resulting disjoint index sets $\{S_i\}_{i=1}^{G}$ define the local matrices $\mathbf{W}_0^{(i)} = \mathbf{W}_0[S_i, :]$.

For FFN blocks, intermediate channels are grouped by the rows of the pretrained up projection. In SwiGLU FFNs, the same groups select rows of `up_proj` and `gate_proj` and columns of `down_proj`, giving row submatrices for the up and gate projections and a column submatrix for the down projection; RoBERTa uses the analogous grouping for its two FFN transformations.

4.2 Local Spectral Characterization and Rank Redistribution

Given these domains, we next decide how the fixed rank budget R is split among them. For each local matrix $\mathbf{W}_0^{(i)}$, truncated randomized SVD (Halko et al., 2011) estimates its leading q_i singular values $\widetilde{\sigma}_{i,j}$; the domain-specific truncation order is given in Appendix A.5. We define

$$E_i^{\mathrm{trunc}}(k) = \frac{\sum_{j=1}^{k} \widetilde{\sigma}_{i,j}^2}{\sum_{j=1}^{q_i} \widetilde{\sigma}_{i,j}^2}, \qquad e_i = \min\left\{k \in \{1, \ldots, q_i\} \,\middle|\, E_i^{\mathrm{trunc}}(k) \geq \tau\right\}. \tag{2}$$

The descriptor e_i counts the directions needed to retain a fraction τ of the energy within the retained spectrum. It differs from the full-spectrum energy-retention rank of Section 3 and depends on the spectral profile, the domain dimensions, and the truncation order; we therefore use it only for relative allocation within a projection.

Let $\bar{e}$ denote the mean descriptor and $\bar{r} = R/G$ the rank under uniform allocation. The relative deviation and the real-valued rank proposal are

$$\gamma_i = \frac{\bar{e} - e_i}{\max_j |\bar{e} - e_j| + \epsilon}, \qquad \widetilde{r}_i = \bar{r}\left[1 + \mathrm{clip}(\lambda\gamma_i, -c, c)\right]. \tag{3}$$

We refer to the deviation $\bar{e} - e_i$ as the *gap*: it measures how far a domain's descriptor departs from the projection-level mean and is unrelated to gaps between adjacent singular values. The coefficient λ

controls redistribution strength, c bounds the relative deviation of the proposals from $\bar{r}$, and rounding followed by budget correction yields integer ranks r_i with $\sum_i r_i = R$ (Appendix A.5).

Under the default policy, domains with smaller descriptors receive larger rank proposals; we call this *low-eRank prioritization* (LRP). The intuition is capacity complementarity—a local mapping whose retained energy occupies few dominant directions may benefit from additional degrees of freedom—but the spectral observations alone do not establish this, so Section 5.4 compares LRP against the uniform and the reversed allocation under the same total budget R.

4.3 Partition-Specific Low-Rank Adaptation

Each domain-rank pair (S_i, r_i) is finally converted into an independent LoRA branch,

$$\Delta \mathbf{W}^{(i)} = \mathbf{B}_i \mathbf{A}_i, \qquad \mathbf{A}_i \in \mathbb{R}^{r_i \times d_{\text{in},i}}, \qquad \mathbf{B}_i \in \mathbb{R}^{d_{\text{out},i} \times r_i}. \tag{4}$$

where $d_{\text{in},i}$ and $d_{\text{out},i}$ are the input and output dimensions of the local matrix: d_{in} and $|S_i|$ for a row partition, and $|S_i|$ and d_{out} for the column partition of the down projection.

For the local decomposition $\mathbf{W}_0^{(i)} = \mathbf{U}_i \mathbf{\Sigma}_i \mathbf{V}_i^\top$, we initialize

$$\mathbf{A}_i = \mathbf{V}_i[:, 1{:}r_i]^\top, \qquad \mathbf{B}_i = \mathbf{0}. \tag{5}$$

This yields a zero initial update while aligning the input-side factor with the dominant local singular directions. Because ranks differ across domains, we set $\alpha_i = \rho r_i$ so that all branches share the same effective scaling ratio. For the attention, up, and gate projections, branch outputs are concatenated along the grouped channel dimension and restored to the original channel order. For the FFN down projection, each branch acts on its own intermediate-channel group and contributes to the shared output space: given the down-projection input $\mathbf{h}$, the adapter contribution is

$$\Delta \mathbf{y}_{\text{down}} = \rho \sum_{i=1}^{G} \mathbf{B}_i^{\text{down}} \mathbf{A}_i^{\text{down}} \mathbf{h}_{S_i}. \tag{6}$$

Because the local updates occupy disjoint rows or columns of their target projection, they can be merged back into that projection after fine-tuning.

During fine-tuning, the pretrained weights, domain assignment, and allocated ranks are fixed; only the local low-rank factors are optimized. GDPLoRA thus provides fine-grained local adaptation without learned routing or token-wise dispatch.

5 Experiments

5.1 Experimental Setup

Datasets and Models. Experiments cover RoBERTa-large (Liu et al., 2019), LLaMA-3-8B, and Qwen3-8B. RoBERTa-large is evaluated on eight GLUE tasks (Wang et al., 2018): CoLA, SST-2, MRPC, QQP, MNLI, QNLI, RTE, and STS-B. LLaMA-3-8B and Qwen3-8B are evaluated on GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021) for mathematical reasoning, MBPP (Austin et al., 2021) and MBPP+ (Liu et al., 2023a) for code generation, and MT-Bench (Zheng et al., 2023) for multi-turn instruction following. MT-Bench responses are scored by Qwen3.7-Max (`qwen3.7-max`) under identical grading prompts and decoding settings for all methods.

Baselines. For RoBERTa-large, we compare against Full FT, LoRA, PiSSA, DoRA, AdaLoRA, RandLoRA (Albert et al., 2025), GraLoRA, and two MoELoRA routing variants. For LLaMA-3-8B and Qwen3-8B, the baselines are LoRA, GraLoRA, and MoLoRA-Top4/Top8.

Implementation Details. PEFT methods are matched to comparable trainable-parameter budgets within each backbone. GDPLoRA uses $G = 4$, $R = 56$, and $\rho = 1.0$ for RoBERTa-large, and backbone-specific (G, R) with $\rho = 2.0$ for LLaMA-3-8B and Qwen3-8B. Unless stated otherwise, results from our pipeline are averaged over three seeds; Full FT updates all parameters within the same pipeline. Configurations and evaluation protocols appear in Appendices A and C.

Table 1: GLUE development-set results on RoBERTa-large. All PEFT methods operate under comparable trainable-parameter budgets. Each result is averaged over three random seeds, and the overall average is computed across the eight reported tasks. Best values per column are in bold.

Method	Trainable Params	CoLA	SST-2	MRPC	MNLI	QNLI	RTE	QQP	STS-B	Avg.
Full FT	~ 355M	0.6796	0.9614	0.9333	0.9020	0.9444	0.8568	**0.8936**	0.9227	0.8867
LoRA	13.44M	0.6940	0.9648	0.9340	0.9046	0.9467	0.8604	0.8891	0.9231	0.8896
PiSSA	13.44M	0.6947	0.9614	0.9356	0.9017	0.9440	0.8628	0.8883	0.9239	0.8891
DoRA	13.22M	0.6973	0.9625	0.9344	0.9052	0.9464	0.8688	0.8890	0.9233	0.8909
AdaLoRA	13.44M	0.6893	0.9614	0.9293	**0.9053**	0.9470	0.8568	0.8784	0.9158	0.8854
RandLoRA	13.88M	0.6922	0.9656	0.9347	0.9023	0.9444	0.8448	0.8914	0.9237	0.8874
GraLoRA	13.44M	0.6876	0.9648	0.9316	0.9038	0.9483	0.8628	0.8927	0.9227	0.8893
MoELoRA-Top1	14.32M	0.6806	0.9618	**0.9369**	0.9035	0.9471	0.8700	0.8879	0.9237	0.8889
MoELoRA-Top4	14.32M	0.6877	0.9622	0.9333	0.9042	**0.9488**	0.8724	0.8884	0.9230	0.8900
GDPLoRA (Ours)	**13.27M**	**0.7040**	**0.9660**	0.9346	0.9048	**0.9488**	**0.8748**	0.8933	**0.9265**	**0.8941**

5.2 Main Results

Results on RoBERTa-Large. Table 1 compares GDPLoRA with existing PEFT methods on the eight GLUE tasks under comparable trainable-parameter budgets. GDPLoRA achieves the highest overall average score of 0.8941 with only 13.27M trainable parameters, and attains the best or tied-best result on five of the eight tasks. Specifically, it surpasses LoRA by 0.45 percentage points in average score and outperforms DoRA—the strongest among the single-adapter baselines—by 0.32 percentage points, while using fewer trainable parameters than both. These results demonstrate that the proposed gradient-decoupled formulation yields consistent gains across diverse linguistic phenomena without increasing the parameter budget.

Results on Large Language Models. Table 2 compares GDPLoRA against LoRA, GraLoRA, and routed MoLoRA variants across two 8B backbones. On LLaMA-3-8B, GDPLoRA achieves the highest score on every reported metric. Compared to LoRA, it improves performance by 2.10 and 3.07 percentage points on the math benchmarks (GSM8K and SVAMP), and by 4.15 and 3.70 percentage points on the code benchmarks (MBPP and MBPP+). These code-generation gains also hold against GraLoRA, with improvements of 0.95 and 0.80 percentage points on the original and extended test sets, respectively. Results on Qwen3-8B show more task-specific variations: while GDPLoRA leads on GSM8K, both code metrics, and the MT-Bench average, it slightly trails MoLoRA-Top8 on SVAMP. Nevertheless, it still outperforms GraLoRA on MBPP and MBPP+ by 0.80 and 1.59 percentage points, respectively. The turn-level MT-Bench scores provide a more granular view of instruction-following capabilities. On LLaMA-3-8B, GDPLoRA improves performance on both turns compared to LoRA, raising the average score from 3.698 to 3.871. On Qwen3-8B, GDPLoRA achieves an average of 5.138, surpassing both LoRA (4.931) and MoLoRA-Top4 (5.056). Although MoLoRA-Top4 attains a marginally higher second-turn score (4.188 vs. 4.175), GDPLoRA's overall advantage is driven by its stronger first-turn performance.

Notably, GDPLoRA achieves these results while using 3.76% fewer trainable parameters than LoRA on LLaMA-3-8B and 0.66% fewer on Qwen3-8B. Its parameter efficiency remains comparable to the MoLoRA variants, using 1.94% fewer parameters on LLaMA-3-8B and 1.14% more on Qwen3-8B. Interestingly, although MoLoRA-Top4 and MoLoRA-Top8 share identical parameter counts, their relative performance reverses across the two backbones: Top8 outperforms Top4 on every LLaMA-3-8B metric, whereas Top4 achieves a higher MT-Bench average on Qwen3-8B. This suggests that merely activating more experts does not reliably yield better performance, and that parameter count alone is insufficient to explain the observed variations in model effectiveness.

5.3 Training Dynamics and Early Stability

We compare the training-loss trajectories of LoRA, MoLoRA, and GDPLoRA on LLaMA-3-8B across MetaMath, code generation, and chat data (Figure 4), using the configurations described in Appendix B. Specifically, LoRA uses $r = 118$ and $\alpha = 236$; MoLoRA-Top8 uses eight experts, each with $r_e = 14$ and $\alpha = 28$; and GDPLoRA uses $G = 8$ and $R = 264$. All loss curves are averaged

Table 2: Results on LLaMA-3-8B and Qwen3-8B under comparable trainable-parameter budgets. T1 and T2 denote the two MT-Bench turns. Results are averaged over three random seeds, and the best value for each backbone is shown in bold. Parameter counts are given in millions; GDPLoRA counts are averaged over the same three seeds, and exact counts are listed in Table 9.

Backbone	Method	Params.	GSM8K	SVAMP	MBPP	MBPP+	MT-Bench		
		(M)					T1	T2	Avg.
LLaMA-3-8B	LoRA	309.330	0.7340	0.7360	0.5960	0.4960	4.625	2.771	3.698
	MoLoRA-Top8	303.563	0.7110	0.7290	0.5840	0.4890	4.178	2.390	3.284
	MoLoRA-Top4	303.563	0.6610	0.6840	0.5430	0.4530	3.942	2.383	3.163
	GraLoRA	309.330	0.7483	0.7567	0.6280	0.5250	4.581	2.706	3.644
	GDPLoRA	297.685	**0.7550**	**0.7667**	**0.6375**	**0.5330**	**4.796**	**2.946**	**3.871**
Qwen3-8B	LoRA	321.896	0.8491	0.8633	0.8100	0.6905	6.062	3.800	4.931
	MoLoRA-Top8	316.146	0.8537	**0.8733**	0.8042	0.6825	5.938	3.562	4.750
	MoLoRA-Top4	316.146	0.8533	0.8600	0.8042	0.6825	5.925	**4.188**	5.056
	GraLoRA	321.896	0.8507	0.8600	0.8042	0.6852	5.763	3.550	4.656
	GDPLoRA	319.762	**0.8571**	0.8567	**0.8122**	**0.7011**	**6.100**	4.175	**5.138**

over the three seeds of the main experiments. Faint lines indicate the unsmoothed step-wise loss with mini-batch fluctuations, while saturated solid curves show the smoothed loss

Across all three panels, GDPLoRA exhibits a smaller initial rise in smoothed loss than MoLoRA-Top8. On MetaMath and code, MoLoRA-Top8 displays a pronounced transient spike before its loss begins to decrease, whereas LoRA and GDPLoRA follow similar trajectories with considerably smaller initial increases. On chat, GDPLoRA remains comparatively steady during the first optimizer steps, while both baselines exhibit larger transients. As optimization progresses, all losses decrease and the inter-method gaps narrow; consequently, the most salient distinction lies in the early transient behaviour rather than in the later convergence trend.

These curves provide qualitative evidence that GDPLoRA yields more stable early optimisation within the displayed window; the downstream task scores reported in Table 2 complement this observation, though the loss trajectories alone do not account for the final performance differences. Two caveats apply. First, because the three methods also differ in adapter architecture and initialisation scheme, this comparison does not isolate the effect of removing routing. Second, because we plot seed-averaged curves, neither mini-batch stochasticity nor seed-to-seed variance is visible, and the displayed window does not establish full-run convergence.

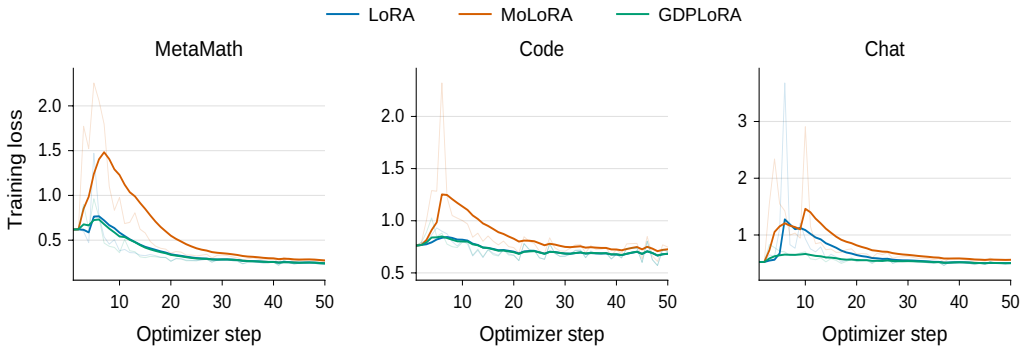


Figure 4: Early training loss trajectories of LoRA, MoLoRA-Top8, and GDPLoRA on LLaMA-3-8B across MetaMath, code, and chat data. Each curve is averaged over three random seeds using the main-experiment configurations. Faint lines show the unsmoothed step-wise loss (retaining mini-batch fluctuations); solid curves show the smoothed loss used for comparison. Within the displayed window, GDPLoRA exhibits smaller initial loss transients than MoLoRA-Top8 on all three tasks.

5.4 Ablation Studies

We conduct ablation studies on RoBERTa-large with a fixed configuration ($R = 64$, $G = 8$) over five GLUE tasks (CoLA, SST-2, MRPC, RTE, and QQP). All reported values are averaged over these five

Table 3: Ablation studies on RoBERTa-large ($R = 64$, $G = 8$), averaged over CoLA, SST-2, MRPC, RTE, and QQP across three random seeds. (a) Effect of replacing each core component of GDPLoRA. (b) Comparison of low-eRank prioritization (LRP) and high-eRank prioritization (HRP) applied independently to the attention (QKVO) and FFN projections.

(a) Core Components

Variant	Avg.
Contiguous Partitioning	0.8711
Uniform Rank	0.8706
w/o SVD Initialization	0.8701
GDPLoRA	**0.8751**

(b) Rank-Allocation Direction

Variant	QKVO	FFN	Avg.
B	HRP	LRP	0.8679
C	LRP	HRP	0.8688
D	HRP	HRP	0.8717
A (GDPLoRA)	**LRP**	**LRP**	**0.8751**

tasks and three random seeds. We first isolate the contributions of three core components—weight-derived partitioning, heterogeneous rank allocation, and domain-specific SVD initialization—and then compare the two rank-allocation directions introduced in Section 4.2.

Core Components. Table 3(a) examines three components by replacing each one individually while keeping the others fixed. Replacing weight-derived domains with contiguous row blocks lowers the average from 0.8751 to 0.8711 (-0.40 points); because both variants retain local branch structures, this difference isolates the effect of the grouping strategy. Replacing heterogeneous rank allocation with uniform ranks (while preserving the same domain partitioning) yields 0.8706 (-0.45), confirming the benefit of non-uniform allocation. Finally, removing domain-specific SVD initialization gives 0.8701 (-0.50). Because each variant modifies exactly one component, these results quantify each component's marginal contribution within the full system; they do not imply that the gains are additive, nor do they establish a definitive ranking of component importance.

Rank-Allocation Direction. Table 3(b) compares low-eRank prioritization (LRP), which assigns larger ranks to domains with smaller effective-rank descriptors e_i, against its reverse, high-eRank prioritization (HRP), applied independently to the attention (QKVO) and FFN projections. The consistent LRP/LRP configuration attains the highest average (0.8751), followed by HRP/HRP (0.8717); both mixed settings underperform (0.8679 and 0.8688), suggesting that consistency between the two module types matters.

The two allocation choices interact: with LRP fixed in the FFN, switching attention from HRP to LRP raises the average by 0.72 points, whereas with HRP fixed in the FFN, the same switch lowers it by 0.29 points. This asymmetry indicates that the two projections should be assessed jointly rather than in isolation. Within the tested configuration ($G = 8$, $R = 64$), the results favor LRP/LRP, though the source of the interaction remains unresolved, and the findings do not establish that smaller descriptors universally imply larger rank requirements.

Hyperparameter Sensitivity. We examine the sensitivity of GDPLoRA to the spectral-energy threshold τ and the redistribution coefficient λ on a four-task subset (CoLA, MRPC, RTE, and SST-2). Across the λ sweep, the average score spans 0.78 points (0.8595–0.8673); across the τ sweep, it spans 1.07 points (0.8593–0.8700). Neither sweep exhibits a monotonic trend, and per-task optima differ, indicating that the aggregate score reflects a trade-off across tasks. Because each sweep fixes the other hyperparameter at its default ($\tau = 0.80$ for the λ sweep, $\lambda = 0.4$ for the τ sweep), neither characterizes the joint neighbourhood of the main configuration $(\tau, \lambda) = (0.85, 0.20)$. Full curves and per-task breakdowns are provided in Appendices B.5 and D.

6 Conclusion

We proposed GDPLoRA, a fine-grained low-rank adaptation method that partitions frozen Transformer projections into local domains and redistributes a fixed per-projection rank budget according to their spectral structure. Domain boundaries and ranks remain static during fine-tuning, so the method needs no learned routing or token-wise dispatch and its adapters merge exactly into the pretrained weights. GDPLoRA improves aggregate performance over competitive PEFT baselines at comparable or lower trainable-parameter budgets. Ablations indicate that weight-derived partitioning, heterogeneous rank allocation, and structured initialization each contribute. A natural next step is to make the local structure task-aware, or to update it during training.

AI USE STATEMENT

In this work, generative AI tools were utilized exclusively for assisting with initial language proofreading and LaTeX template formatting. All core conceptual ideas, theoretical derivations, algorithm designs, code implementations, and experimental formulations were independently conceived, verified, and authored by the researchers. The authors take full responsibility for the entirety of the content, claims, and formulations presented in this paper.

ETHICS STATEMENT

This research focuses on foundational machine learning algorithms. All datasets used are standard, publicly available academic benchmarks. The proposed method does not generate harmful content, involve sensitive personal data, or introduce societal bias.

REPRODUCIBILITY STATEMENT

Section 4 specifies the partitioning, local effective-rank estimation, rank allocation, initialization, and forward computation used by GDPLoRA. The main experimental settings are reported in Section 5, with model-specific hyperparameters, exact trainable-parameter counts, random seeds, and evaluation details provided in the appendix and supplementary material.

REFERENCES

Paul Albert, Frederic Z. Zhang, Hemanth Saratchandran, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. RandLoRA: Full rank parameter-efficient fine-tuning of large models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 22258–22270, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/382b95a6580cfb0b5ca33c74b4e0e770-Paper-Conference.pdf.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, may 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581/.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. Analyzing redundancy in pretrained transformer models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4908–4926, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.398. URL https://aclanthology.org/2020.emnlp-main.398/.

Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1-2):143–175, 2001. ISSN 1573-0565. doi: 10.1023/a:1007612920971. URL https://doi.org/10.1023/a:1007612920971.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.106. URL https://aclanthology.org/2024.acl-long.106/.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3/.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. ISSN 1095-7200. doi: 10.1137/090771806. URL https://doi.org/10.1137/090771806.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yeonjoon Jung, Daehyun Ahn, Hyungjun Kim, Taesu Kim, and Eunhyeok Park. GraLoRA: Granular low-rank adaptation for parameter-efficient fine-tuning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 95822–95845. Curran Associates, Inc., 2025. doi: 10.52202/085713-3205. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/89f8c6687c4adafe4d500444cd3db39a-Paper-Conference.pdf.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. MixLoRA: Enhancing large language models fine-tuning with LoRA-based mixture of experts. arXiv preprint arXiv:2404.15159, 2024. URL https://arxiv.org/abs/2404.15159.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. SVFT: Parameter-efficient fine-tuning with singular vectors. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 41425–41446. Curran Associates, Inc., 2024. doi: 10.52202/079017-1311. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/48c368f105e8145b945227b73255635a-Paper-Conference.pdf.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023a. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/43e9d647ccd3e4b7b5baab53f0368686-Paper-Conference.pdf.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. When MOE meets LLMs: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024, pp. 1104–1114. Association for Computing Machinery, July 2024a. doi: 10.1145/3626772.3657722. URL https://doi.org/10.1145/3626772.3657722.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/liu24bn.html.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692, 2019. URL https://arxiv.org/abs/1907.11692.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. Deja vu: Contextual sparsity for efficient LLMs at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22137–22176. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/liu23am.html.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 121038–121072. Curran Associates, Inc., 2024. doi: 10.52202/079017-3846. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/db36f4d603cc9e3a2a5e10b93e6428f2-Paper-Conference.pdf.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, pp. 2080–2094, Online, jun 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Shaomu Tan, Di Wu, and Christof Monz. Neuron specialization: Leveraging intrinsic task modularity for multilingual machine translation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6506–6527, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.374. URL https://aclanthology.org/2024.emnlp-main.374/.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, Dubrovnik, Croatia, may 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.239. URL https://aclanthology.org/2023.eacl-main.239/.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580/.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, nov 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. Finding skill neurons in pre-trained transformer-based language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11132–11152, Abu Dhabi, United Arab Emirates, dec 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.765. URL https://aclanthology.org/2022.emnlp-main.765/.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of LoRA experts. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 47302–47318, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/ce806d8b4bf38cd92d483e5a0490d983-Paper-Conference.pdf.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 30745–30766, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/82eec786fdfbbfa53450c5feb7d1ac92-Paper-Conference.pdf.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 71768–71791. Curran Associates, Inc., 2024. doi: 10.52202/079017-2292. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/83f95bb0ac5046338ea2afe3390e9f4b-Paper-Conference.pdf.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical

questions for large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 45040–45061, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/c400474e8a36d0812fdee52739288b12-Paper-Conference.pdf.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient MoE for instruction tuning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 25401–25420, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6d00071564ec447466fc4577743cf1b3-Paper-Conference.pdf.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=lq62uWRJjiY.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. MoEfication: Transformer feed-forward layers are mixtures of experts. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.71. URL https://aclanthology.org/2022.findings-acl.71/.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Emergent modularity in pre-trained transformers. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4066–4083, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.250. URL https://aclanthology.org/2023.findings-acl.250/.

Zhengyan Zhang, Chaojun Xiao, Qiujieli Qin, Yankai Lin, Zhiyuan Zeng, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Exploring the benefit of activation sparsity in pre-training. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60040–60056. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhang24bq.html.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. OpenCodeInterpreter: Integrating code generation with execution and refinement. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12834–12859, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.762. URL https://aclanthology.org/2024.findings-acl.762/.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15913–15923, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.890. URL https://aclanthology.org/2024.emnlp-main.890/.

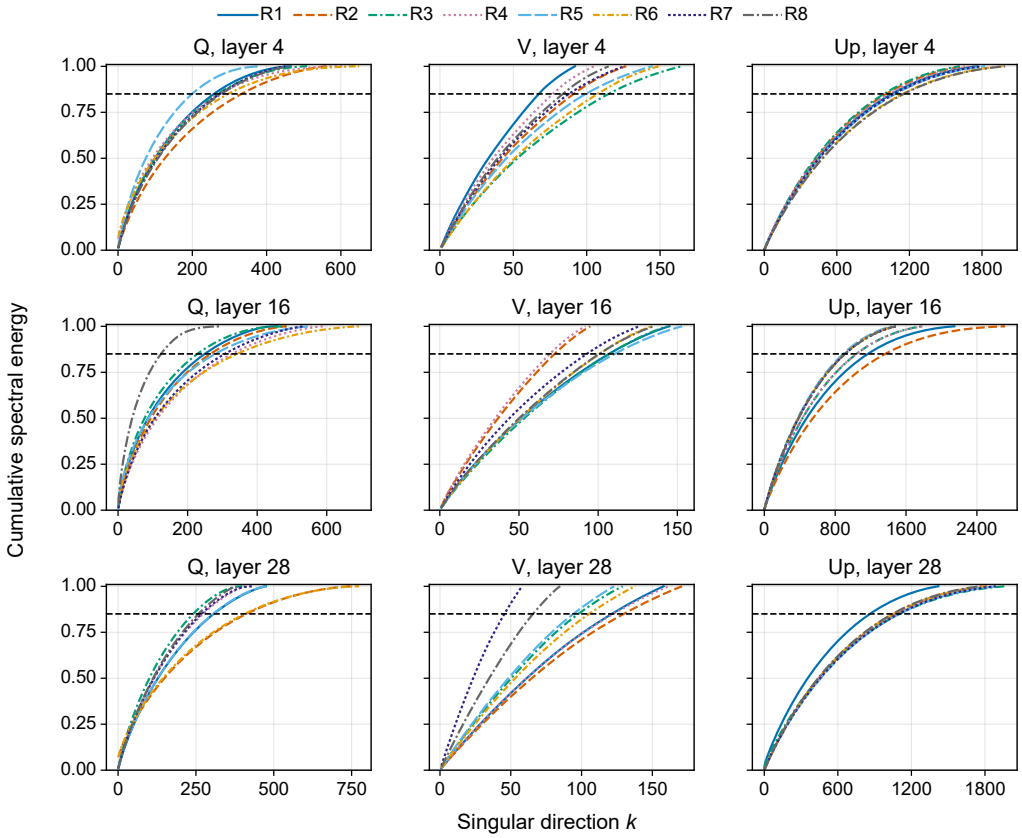


Figure 5: Representative cumulative spectral-energy profiles of weight-derived local regions in frozen LLaMA-3-8B. Rows correspond to representative Transformer layers (Layers 4, 16, and 28), while columns show the `q_proj`, `v_proj`, and `up_proj` projections. Each curve denotes one local region. The different accumulation rates within the same projection illustrate local variation in spectral concentration across early, middle, and late layers.

.1 Additional Local Spectral Observations

Figure 5 provides additional examples of the cumulative spectral-energy profiles of weight-derived local regions in frozen LLaMA-3-8B. We show representative early, middle, and late Transformer layers together with attention and FFN projections. Across these settings, local regions within the same projection exhibit visibly different rates of spectral-energy accumulation, showing that the observed local spectral variation is not restricted to a single layer or projection.

.2 Partition-Specific Rank Allocation

Figures 6 and 7 visualize the local ranks assigned by GDPLoRA on LLaMA-3-8B. The FFN results show the partition-specific rank assignments across Transformer layers, while the attention results provide the corresponding allocations for the Q, K, V, and O projections.

A Additional Implementation Details

A.1 RoBERTa-Large Experimental Setup

All parameter-efficient fine-tuning experiments use the publicly available `roberta-large` checkpoint. We evaluate eight GLUE tasks: CoLA, SST-2, MRPC, QQP, MNLI, QNLI, RTE, and STS-B. Adaptation is applied to the query, key, value, and output projections and to the two FFN linear

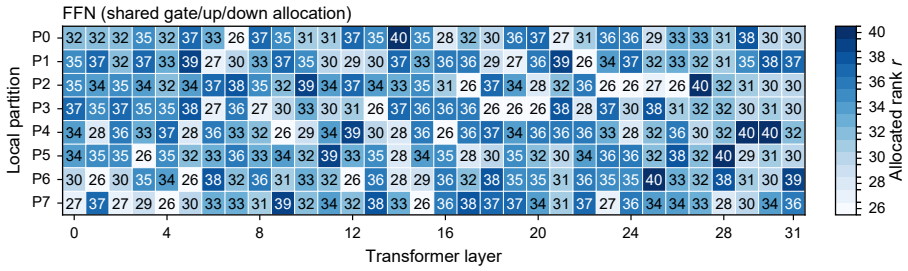


Figure 6: Partition-specific FFN rank allocations produced by GDPLoRA on LLaMA-3-8B. Each column corresponds to a Transformer layer, and each cell indicates the rank assigned to one local FFN domain. Although the total rank budget of each target matrix is fixed, different local domains receive different adaptation ranks. Partition indices are local to each target matrix and do not imply cross-layer correspondence.

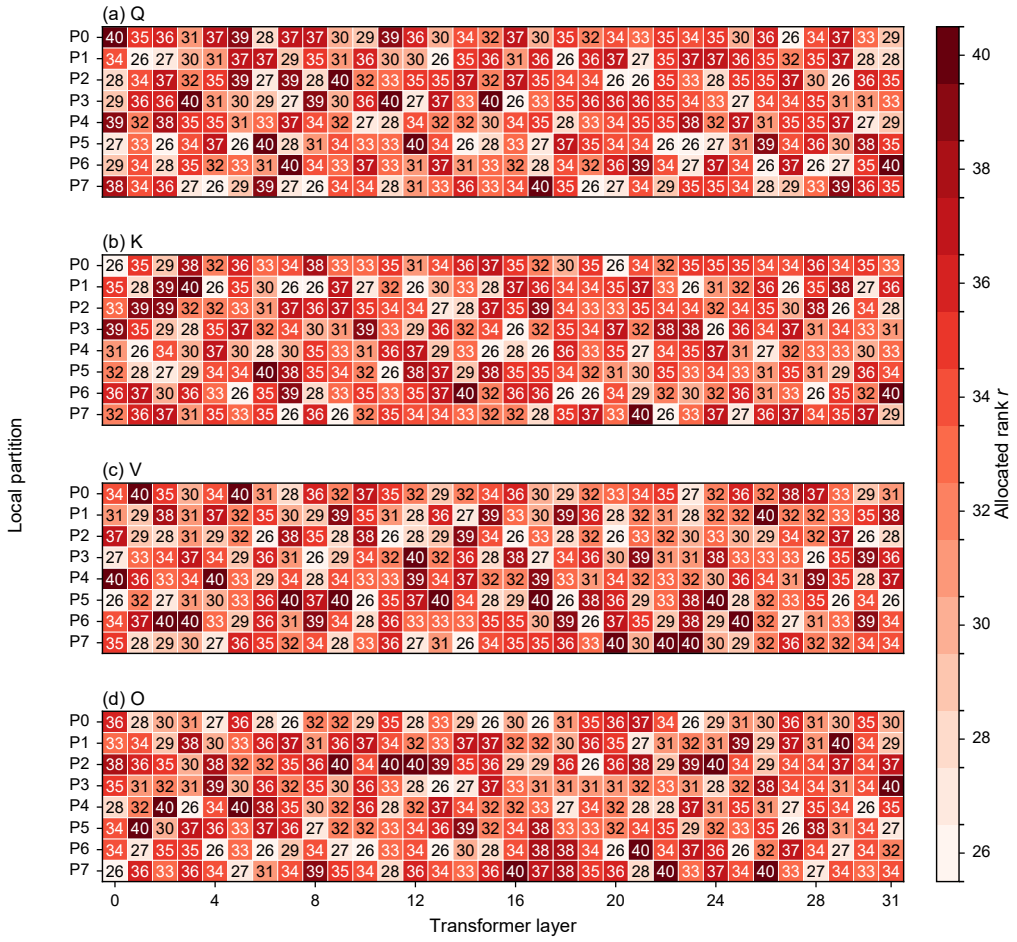


Figure 7: Partition-specific rank allocations for the attention projections of LLaMA-3-8B. The panels correspond to the Q, K, V, and O projections across all Transformer layers. Each cell indicates the rank assigned to one local partition. Partition indices are local to each target matrix and do not imply cross-layer correspondence.

transformations in every encoder layer. The pretrained backbone remains frozen, while the low-rank parameters and the sequence-classification head are optimized.

Table 4: Shared task-level configurations for RoBERTa-large.

Task	Epochs	Batch Size	Max Len.
CoLA	30	32	128
SST-2	10	32	128
MRPC	30	32	128
MNLI	5	32	128
QNLI	10	32	128
RTE	30	32	128
QQP	5	32	128
STS-B	10	32	128

The Full FT results in Table 1 are obtained in our experiments by updating all model parameters and are averaged over three random seeds. The results are reported on the $[0, 1]$ scale used throughout the paper. Controlled comparisons are made primarily among PEFT methods trained under the same implementation and evaluation settings.

A.2 GLUE Data Processing and Metrics

GLUE is loaded through Hugging Face Datasets using the `roberta-large` tokenizer. All inputs are padded or truncated to a maximum sequence length of 128. CoLA and SST-2 use single-sentence inputs, while MRPC, QQP, MNLI, QNLI, RTE, and STS-B use their standard sentence-pair formats. MNLI is evaluated on `validation_mismatched`; the remaining tasks use their standard validation splits.

We report Matthews correlation coefficient for CoLA, F1 for MRPC and QQP, correlation for STS-B, and classification accuracy for SST-2, MNLI, QNLI, and RTE. The average GLUE score is the arithmetic mean of the eight task-level values after representing all metrics on the $[0, 1]$ scale.

A.3 RoBERTa-Large Optimization and Method Configurations

All independently trained PEFT methods use AdamW (`adamw_torch`) with cosine learning-rate decay, a warmup ratio of 0.03, zero weight decay, and a maximum gradient norm of 1.0. Training uses FP16 with a per-device batch size of 32 and one gradient-accumulation step. Evaluation and checkpoint saving are performed after every epoch, and the checkpoint with the best validation metric is used for final evaluation.

LoRA, PiSSA, DoRA, and AdaLoRA are implemented using the Hugging Face PEFT library. MoELoRA and GDPLoRA use custom PyTorch implementations, while GraLoRA and RandLoRA follow their corresponding parameterizations under the same training and evaluation pipeline. Standard LoRA and PiSSA use rank $r = 28$, DoRA uses $r = 27$, AdaLoRA reduces its adaptive rank budget from 42 toward 28, RandLoRA uses $r = 12$, and GraLoRA uses $r = 28$ with granularity $k = 2$. MoELoRA is evaluated with Top-1 and Top-4 routing.

For the main RoBERTa-large experiments, GDPLoRA uses $G = 4$ local domains and a per-projection rank budget of $R = 56$. The spectral-energy threshold is $\tau = 0.85$, the redistribution coefficient is $\lambda = 0.20$, and the shared scaling ratio is

$$\rho = \frac{\alpha_i}{r_i} = 1.0. \tag{7}$$

The ablation experiments use a separate configuration with $G = 8$ and $R = 64$. This configuration is used for the controlled component and rank-allocation studies in Section 5.4.

Table 5: Primary RoBERTa-large PEFT configurations.

Method	Rank	Groups	Configuration
LoRA	$r = 28$	–	standard LoRA
PiSSA	$r = 28$	–	`pissa_niter_4`
DoRA	$r = 27$	–	PEFT implementation
AdaLoRA	$42 \rightarrow 28$	–	adaptive rank
RandLoRA	$r = 12$	–	random basis
GraLoRA	$r = 28$	$k = 2$	granular partition
MoELoRA	$r = 28$	–	Top-1 / Top-4
GDPLoRA	$R = 56$	$G = 4$	$\tau = .85, \lambda = .20, \rho = 1.0$

Table 6: Task-specific learning rates for RoBERTa-large.

Method	CoLA	SST-2	MRPC	MNLI	QNLI	RTE	QQP	STS-B
LoRA	$1e{-4}$	$2e{-4}$	$1e{-4}$	$1e{-4}$	$2e{-4}$	$2e{-4}$	$1e{-4}$	$2e{-4}$
PiSSA	$4e{-5}$	$5e{-5}$	$5e{-5}$	$4e{-5}$	$4e{-5}$	$5e{-5}$	$5e{-5}$	$5e{-5}$
DoRA	$2e{-4}$	$2e{-4}$	$2e{-4}$	$1e{-4}$	$2e{-4}$	$1e{-4}$	$1e{-4}$	$2e{-4}$
GraLoRA	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$
RandLoRA	$1e{-4}$	$1e{-4}$	$2e{-4}$	$1e{-4}$	$1e{-4}$	$5e{-5}$	$1e{-4}$	$1e{-4}$
AdaLoRA	$2e{-4}$	$2e{-4}$	$2e{-4}$	$1e{-4}$	$2e{-4}$	$3e{-4}$	$3e{-4}$	$3e{-4}$
MoELoRA	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$	$2e{-4}$
GDPLoRA	$2e{-4}$	$2e{-4}$	$3e{-4}$	$1e{-4}$	$1e{-4}$	$1e{-4}$	$2e{-4}$	$1e{-4}$

A.4 RoBERTa-Large Task-Specific Learning Rates

A.5 GDPLoRA Construction Details

Local partitioning. For attention projections and FFN up projections, the output-row vectors used to form the groups are first normalized,

$$\mathbf{x}_j = \frac{\mathbf{W}_0[j,:]}{\|\mathbf{W}_0[j,:]\|_2}, \tag{8}$$

and partitioned using size-regularized cosine K-Means. At each assignment step,

$$z_j = \arg\min_c \left(1 - \mathbf{x}_j^\top \boldsymbol{\mu}_c + \eta \frac{n_c}{N}\right), \tag{9}$$

where $\boldsymbol{\mu}_c$ is the normalized centroid of domain c, n_c is its current size, N is the number of rows, and $\eta = 0.02$. The occupancy term discourages highly imbalanced domains but does not enforce equal-sized partitions. Centroids are initialized from randomly selected normalized rows, and clustering runs for at most 20 alternating iterations.

For Q/K/V/O projections, clustering is applied independently to their output rows. In RoBERTa FFNs, intermediate-projection rows are clustered and the corresponding output-projection columns follow the same ordering. For LLaMA-3-8B and Qwen3-8B, clustering is performed on `up_proj` rows, with the same ordering applied to `gate_proj` and `up_proj` rows and to `down_proj` columns. This permutation preserves the frozen FFN function before adaptation.

Truncated spectral estimation. For each local matrix $\mathbf{W}_0^{(i)} \in \mathbb{R}^{d_{\text{out},i} \times d_{\text{in},i}}$, the implementation applies `torch.svd_lowrank` to $(\mathbf{W}_0^{(i)})^\top$ in FP32. The truncation order is

$$\begin{aligned} q_{\max,i} &= \min(d_{\text{out},i}, d_{\text{in},i}) - 1, \\ q_i &= \max\left(4,\ \min\left[q_{\max,i},\ \left\lfloor \frac{\max(d_{\text{out},i}, d_{\text{in},i})}{4} \right\rfloor\right]\right). \end{aligned} \tag{10}$$

with safeguards for very small regions. The local spectral descriptor is therefore normalized by the retained approximate spectral energy, rather than by the full matrix energy. The full FP32 Gram

spectra used for the motivational diagnostic are a separate measurement. We do not assume that the full-spectrum and truncated descriptors have identical values or domain rankings.

Adapter initialization uses a separate truncation order

$$q_{\text{init}} = \max\left(1,\ \min(r_i + 10, d_{\text{in},i}, d_{\text{out},i})\right). \tag{11}$$

The input-side factor is initialized from the leading r_i singular directions, while the output-side factor is initialized to zero. The initial adapter update is therefore exactly zero.

Integer rank-budget correction. The proposal $\widetilde{r}_i$ in Eq. 3 is real-valued. Each domain is first assigned $\widehat{r}_i = \max\{1, \text{round}(\widetilde{r}_i)\}$, and we initialize the corrected ranks as $r_i = \widehat{r}_i$. The remaining budget discrepancy is $\delta = R - \sum_i r_i$. If $\delta > 0$, ranks are added according to the largest residual $\widetilde{r}_i - r_i$. If $\delta < 0$, ranks are removed from eligible domains according to the smallest residual, while retaining $r_i \geq 1$. The real-valued proposal is retained when computing these residuals. The final allocation satisfies

$$\sum_{i=1}^{G} r_i = R \tag{12}$$

for every adapted attention projection and FFN block.

Connection to initial adaptation directions. Consider the local linear branch without adapter dropout, $\mathbf{W}_{\text{eff}}^{(i)} = \mathbf{W}_0^{(i)} + \rho \mathbf{B}_i \mathbf{A}_i$. Let $\mathbf{V}_{i,r_i}$ contain the retained local right singular directions used to initialize $\mathbf{A}_i$, so that $\mathbf{A}_i^{(0)} = \mathbf{V}_{i,r_i}^\top$ and $\mathbf{B}_i^{(0)} = \mathbf{0}$. Writing $\mathbf{D}_i$ for the loss gradient with respect to $\mathbf{W}_{\text{eff}}^{(i)}$ at initialization, the chain rule gives

$$\nabla_{\mathbf{B}_i} \mathcal{L}|_0 = \rho \mathbf{D}_i \mathbf{V}_{i,r_i}, \qquad \nabla_{\mathbf{A}_i} \mathcal{L}|_0 = \mathbf{0}.$$

The pretrained singular directions therefore determine how the task gradient initially enters the adapter, while r_i determines the number of retained directions. This provides an analytic connection between pretrained spectral structure and the initial adaptation geometry. The task gradient $\mathbf{D}_i$ is not determined by pretrained singular values, so the identity does not establish which domains benefit most from additional rank. It is not a characterization of the full AdamW training dynamics.

B LLM Training Details

B.1 Models and Target Modules

We evaluate LLaMA-3-8B and Qwen3-8B. Low-rank adaptation is applied to the attention projections `q_proj`, `k_proj`, `v_proj`, and `o_proj`, and to the SwiGLU projections `gate_proj`, `up_proj`, and `down_proj` in every decoder layer. All pretrained backbone parameters remain frozen.

B.2 Training Datasets and Optimization

Separate supervised fine-tuning runs are conducted for mathematical reasoning, code generation, and instruction following. Mathematical reasoning uses `meta-math/MetaMathQA` (Yu et al., 2024), mapping `query` to `response`; code generation uses `m-a-p/CodeFeedback-Filtered-Instruction` (Zheng et al., 2024), mapping `query` to `answer`; and instruction following uses `WizardLM/WizardLM_evol_instruct_70k` (Xu et al., 2024), mapping `instruction` to `output`. Each configuration is trained for one epoch.

Prompt tokens are masked with -100, and the loss is computed only over response tokens. Both backbones use AdamW with cosine learning-rate decay, a warmup ratio of 0.03, zero weight decay, a maximum gradient norm of 1.0, and zero adapter dropout. Gradient checkpointing is enabled. LLaMA-3-8B uses a learning rate of 5×10^{-4}, while Qwen3-8B uses 3×10^{-4}. Frozen backbone weights are maintained in BF16, while adapter parameters are optimized in FP32.

Table 7: Training datasets for the 8B language models.

Subtask	Dataset	Samples	Max Len.
Math	MetaMathQA	100,000	512
Code	CodeFeedback	100,000	1024
Chat	WizardLM	20,000	2048

Table 8: Main GDPLoRA configurations for the evaluated backbones.

Backbone	G	R	τ	λ	ρ
RoBERTa-large	4	56	0.85	0.20	1.0
LLaMA-3-8B	8	264	0.90	0.20	2.0
Qwen3-8B	4	220	0.90	0.20	2.0

B.3 Method-Specific Configurations

For both LLaMA-3-8B and Qwen3-8B, standard LoRA uses rank $r = 118$ and $\alpha = 236$. GraLoRA uses $r = 118$, partition granularity $k = 2$, $\alpha = 236$, and `hybrid_r=0`. MoLoRA uses $E = 8$ matrix-wide routed experts, with rank $r_e = 14$ and $\alpha = 28$ per expert, and is evaluated with Top-4 and Top-8 routing.

For LLaMA-3-8B, GDPLoRA uses $G = 8$ static local domains and a per-core rank budget of $R = 264$, with spectral-energy threshold $\tau = 0.90$, redistribution coefficient $\lambda = 0.20$, and shared scaling ratio $\rho = 2.0$. For Qwen3-8B, GDPLoRA uses $G = 4$, $R = 220$, $\tau = 0.90$, $\lambda = 0.20$, and $\rho = 2.0$. The same target-module set and frozen-backbone protocol are used across compared methods.

The implementation settings `K` and `TOTAL_R_BUDGET` correspond to G and R in the paper. Here, R denotes the rank budget assigned independently to each adapted attention projection or FFN core rather than a model-wide rank budget. For GDPLoRA, the resulting budget is redistributed among the G local domains according to their pretrained spectral descriptors, while MoLoRA assigns rank $r_e = 14$ independently to each routed expert.

B.4 Trainable-Parameter Accounting

All pretrained projection weights and biases remain frozen. Trainable-parameter counts include all low-rank factors and, for routed MoE-LoRA baselines, router parameters. A branch with local input size $d_{\text{in},i}$, local output size $d_{\text{out},i}$, and rank r_i contributes

$$P_i = r_i(d_{\text{in},i} + d_{\text{out},i}) \tag{13}$$

trainable parameters. For a row partition this becomes $r_i(d_{\text{in}} + |S_i|)$; for a down-projection column partition it becomes $r_i(|S_i| + d_{\text{out}})$. Fixing $\sum_i r_i = R$ preserves the rank budget, while the exact parameter count also depends on the domain sizes and allocated ranks.

Table 9 lists the parameter counts for the LLM configurations. Table 2 reports the same counts rounded to three decimal places in millions. MoLoRA-Top4 and MoLoRA-Top8 share the same total number of trainable parameters. GDPLoRA counts are averaged over the three random seeds used for the main results, since domain sizes and allocated ranks can vary across seeds.

B.5 Hyperparameter Sensitivity

Figure 8 reports the average performance under different values of λ and τ, while Table 10 provides the corresponding numerical results.

Table 9: Trainable-parameter counts for the LLM configurations. GDPLoRA counts are means over the three random seeds used for the main results; the baseline counts are fixed by their configurations.

Method	**LLaMA-3-8B**	**Qwen3-8B**
LoRA	309,329,920	321,896,448
GraLoRA	309,329,920	321,896,448
GDPLoRA	297,685,047	319,762,186
MoLoRA-Top8	303,562,752	316,145,664
MoLoRA-Top4	303,562,752	316,145,664

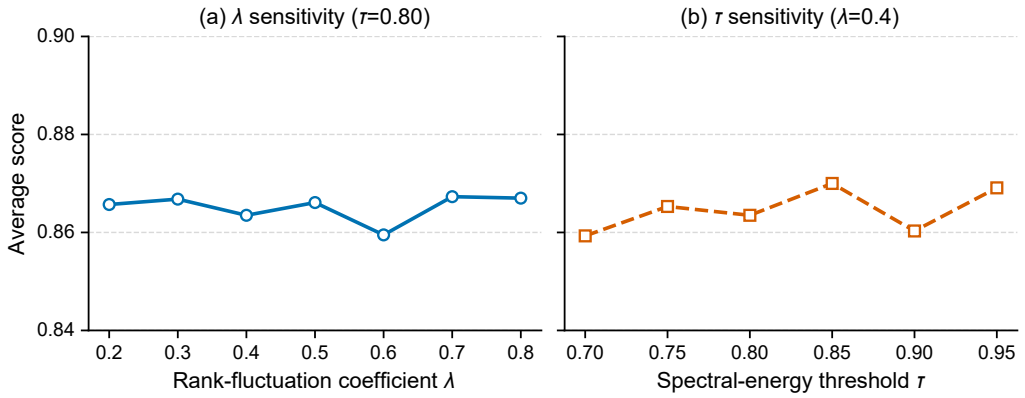


Figure 8: Sensitivity of the average GLUE score to the rank-redistribution coefficient λ and spectral-energy threshold τ.

C Evaluation Protocols

C.1 GLUE

GLUE preprocessing, validation splits, and task-specific metrics are described in Appendix A.2. Each reported result uses the checkpoint selected by the corresponding validation metric.

C.2 GSM8K and SVAMP

GSM8K is evaluated on the official `openai/gsm8k` test split using exact-match accuracy. The final number in each reference answer is treated as the gold answer, while the final number in the generated response is treated as the prediction. Formatting characters such as commas, spaces, percentage symbols, and trailing punctuation are removed before comparison.

SVAMP is evaluated on its test split. The problem body and question are concatenated to form the model instruction, and the final number in the generated response is compared with the reference answer using a numerical tolerance of 10^{-3}. Both benchmarks use deterministic decoding with temperature 0 and at most 512 newly generated tokens.

C.3 MBPP and MBPP+

We generate one completion per problem and report pass@1. Problems and test cases are loaded through the official EvalPlus interface. Generated programs are sanitized using the EvalPlus utility before evaluation. The evaluator reports performance on both the original MBPP tests and the extended MBPP+ tests. Generation uses temperature 0 and at most 512 newly generated tokens.

Table 10: Results for the rank-redistribution coefficient λ, with $\tau = 0.80$. Bold denotes the highest average.

λ	CoLA	MRPC	RTE	SST-2	Avg.
0.2	0.6927	0.9317	0.8773	0.9610	0.8657
0.3	0.6917	0.9300	0.8845	0.9610	0.8668
0.4	0.6812	0.9284	0.8809	0.9633	0.8635
0.5	0.6773	0.9293	0.8989	0.9587	0.8661
0.6	0.6774	0.9331	0.8664	0.9610	0.8595
0.7	**0.7053**	0.9305	0.8700	**0.9633**	**0.8673**
0.8	0.6995	0.9231	0.8845	0.9610	0.8670

Table 11: Results for the spectral-energy threshold τ, with $\lambda = 0.4$. Bold denotes the highest average.

τ	CoLA	MRPC	RTE	SST-2	Avg.
0.70	0.6919	0.9309	0.8556	0.9587	0.8593
0.75	0.6961	0.9278	0.8809	0.9564	0.8653
0.80	0.6812	0.9284	0.8809	0.9633	0.8635
0.85	**0.7021**	0.9301	0.8845	0.9633	**0.8700**
0.90	0.6751	0.9317	0.8700	**0.9644**	0.8603
0.95	0.6775	**0.9329**	**0.9025**	0.9633	0.8691

C.4 MT-Bench

MT-Bench responses are generated sequentially for the two turns of each question. The first-turn interaction is included in the context for the second turn. Response generation uses temperature 0.7 and at most 512 newly generated tokens.

All compared methods are judged using Qwen3.7-Max (`qwen3.7-max`) under identical grading prompts and inference settings. We report the mean Turn-1 score, mean Turn-2 score, and their arithmetic average.

D Tabulated Hyperparameter Sensitivity Results

This section reports the task-level values corresponding to the sensitivity curves in Figure 8. Each average is computed over CoLA, MRPC, RTE, and SST-2 after representing all metrics on the $[0, 1]$ scale. The λ sweep fixes $\tau = 0.80$, whereas the τ sweep fixes $\lambda = 0.4$.

The λ sweep yields average scores between 0.8595 and 0.8673, with the highest mean at $\lambda = 0.7$ when $\tau = 0.80$. The τ sweep yields scores between 0.8593 and 0.8700, with the highest mean at $\tau = 0.85$ when $\lambda = 0.4$. The intermediate decreases in both sweeps show that increasing either value does not reliably improve the aggregate result.

The task-level values also reveal different preferences. In the τ sweep, CoLA favors 0.85, whereas RTE favors 0.95. RTE spans 4.69 percentage points across this sweep, compared with 0.80 percentage points for SST-2. The average therefore masks larger changes on some individual tasks. These results characterize the tested one-dimensional sweeps and support task-aware validation of the hyperparameters, without identifying a universally preferred setting.

E Hardware and Software Environment

The LLaMA-3-8B and Qwen3-8B experiments were conducted on NVIDIA A800 80GB PCIe GPUs, while the RoBERTa-large experiments, including the main GLUE comparison, ablation studies, hyperparameter sensitivity experiments, and other analysis experiments, were conducted on NVIDIA GeForce RTX 3090 GPUs. Methods compared within the same backbone used the same hardware,

Table 12: Hardware configurations used in the experiments.

Experiments	GPU	GPUs/Run
LLaMA-3-8B / Qwen3-8B	A800 80GB PCIe	4
RoBERTa-large	RTX 3090	1
Ablation / Sensitivity	RTX 3090	1
Other analyses	RTX 3090	1

Table 13: Software environments used for the A800-based LLM experiments and RTX 3090-based experiments.

Component	A800	RTX 3090
NVIDIA Driver	595.84	550.163.01
CUDA Compatibility	13.3	12.4
PyTorch CUDA	12.1	12.4
Python	3.10.20	3.10.20
PyTorch	2.5.1+`cu121`	2.6.0+`cu124`
cuDNN	9.1.0	9.1.0
Transformers	5.5.4	5.8.1
Datasets	4.8.4	4.8.5
PEFT	0.10.0	0.19.1

software, data-processing, training, and evaluation settings. The hardware and software configurations are summarized in Tables 12 and 13.