GARD: Gradient-Spectral Rank Allocation for LoRA Fine-Tuning

Preprint. Under review.

Anonymous

Abstract

Parameter-efficient fine-tuning is now the standard approach for adapting large language models, yet most Low-Rank Adaptation (LoRA) configurations still rely on a single uniform rank across layers, ignoring substantial variation in layer-wise sensitivity and gradient structure. This uniformity can waste adapter parameters in some layers while constraining others, especially under strict memory and latency budgets. To address this mismatch, we introduce **GARD**, a gradient-aware rank allocation framework that uses layer-wise gradient spectral analysis to distribute LoRA capacity under a global parameter budget. GARD estimates per-layer gradient covariance spectra from mini-batch gradients, derives an effective dimensionality signal, and maps it to discrete LoRA ranks while enforcing a fixed adapter parameter budget; a dynamic variant periodically updates ranks during training using smoothed spectra. Conceptually, this links intrinsic gradient dimensionality to low-rank adapter capacity, offering a principled alternative to uniform or heuristic rank choices. Empirically, we execute a single GARD-controlled configuration for language model fine-tuning and report a validation perplexity of 17.5529 with a mean LoRA parameter count of 2,570.0. These results demonstrate that gradient-aware rank allocation can be instantiated end-to-end under a tight adapter budget, and they establish an empirical reference point for future multi-seed, multi-baseline evaluations of gradient-spectral capacity allocation in parameter-efficient fine-tuning.

1 Introduction

The rapid adoption of large language models (LLMs) has made fine-tuning under strict memory and compute constraints a central challenge for both research and deployment. Full-parameter fine-tuning of models with billions of parameters is often infeasible in practical settings, motivating a rich line of parameter-efficient fine-tuning (PEFT) methods such as adapters, prefix and prompt tuning, and low-rank adaptation. Among these, Low-Rank Adaptation (LoRA) has emerged as a particularly attractive approach for LLMs, injecting trainable low-rank matrices into selected weight matrices while keeping the original weights frozen. In practice, LoRA is now integrated into widely used toolchains and has been applied across domains from code generation and medical imaging to protein modeling. Despite this success, the dominant configuration pattern remains simple: a single uniform rank is chosen and applied to every adapted layer, even though empirical analyses indicate that transformer layers contribute unevenly to downstream performance.

Uniform rank allocation is at odds with the heterogeneous roles and sensitivities of different layers in transformer-based LLMs. Empirical studies of transformer internals suggest that attention and feedforward blocks at different depths capture distinct linguistic and task-specific phenomena, and that gradients and activations exhibit varying degrees of low-rank structure across layers and tasks. Work on model compression and low-rank approximation has repeatedly observed that both representations and optimization landscapes are effectively low-dimensional but with layer- and task-dependent structure. When LoRA uses a fixed rank everywhere, some layers are over-parameterized, consuming adapter parameters without commensurate gains, while others are under-parameterized,

limiting downstream performance, especially under tight adapter budgets. This mismatch becomes particularly salient when practitioners target strict adapter footprints for deployment on constrained hardware.

Recent PEFT research has begun to question this uniformity by introducing selective adaptation, adaptive budgets, and dynamic structures. Methods such as AdaLoRA allocate rank based on parameter importance and budget constraints [14], DyLoRA varies low-rank structure over training [10], and DoRA explores dynamic rank distribution to better match layer needs [7]. Other works investigate sparsity in PEFT [6], mixture-of-adapter structures [13], or structure-aware low-rank adaptation. However, these approaches typically rely on magnitude-based signals, architectural heuristics, or search over discrete design spaces [4, 5]. What remains missing is a principled, gradient-aware mechanism that ties the intrinsic dimensionality of layer-wise updates to the allocation of low-rank capacity under an explicit global parameter budget.

Building on observations that gradients and Hessians in deep networks often exhibit rapidly decaying spectra and low effective rank, we propose to use gradient spectral information as a first-class signal for LoRA capacity allocation. Gradient covariance spectra summarize not only the magnitude but also the diversity of update directions a layer experiences during fine-tuning. Layers whose gradients span many significant directions plausibly require higher-rank adapters to capture task-specific changes, whereas layers whose gradients are effectively low-rank can be modeled with smaller adapters without sacrificing expressivity. This perspective complements existing PEFT heuristics by grounding rank allocation in the structure of the optimization signals driving adaptation and by connecting PEFT design to dynamical low-rank approximation frameworks developed in numerical analysis.

This paper introduces **GARD** (Gradient-Aware Rank Distribution), a framework that uses layer-wise gradient spectral analysis to allocate LoRA ranks under a global trainable-parameter budget. GARD first estimates, for each adapted weight matrix, an approximate gradient covariance matrix from mini-batch gradients collected during a short profiling phase or online during training. From the eigenvalue spectrum of this covariance, it derives an effective rank or energy-based importance score that reflects how many independent update directions the layer exhibits. These scores are then mapped to discrete LoRA ranks via a budget-constrained allocation procedure that enforces a fixed total number of adapter parameters. A static variant performs this procedure once before fine-tuning, yielding a fixed non-uniform rank profile, while a dynamic variant periodically updates ranks based on smoothed spectra, allowing capacity to flow between layers as training progresses.

The contributions of this work are threefold.

- First, we formalize layer-wise gradient spectral analysis as a practical signal for LoRA capacity allocation, providing approximations suitable for billion-parameter transformers and integrating them into an explicit budget-constrained optimization framework.
- Second, we introduce GARD, encompassing both static and dynamic gradient-aware rank allocation schemes that redistribute LoRA rank across layers while respecting a global adapter parameter budget, and we describe implementation strategies for stable rank changes over the course of training.
- Third, we provide an initial empirical instantiation of GARD in language model fine-tuning, reporting results from a single configuration that operates under a strict adapter budget and establishing a concrete reference point for future comparisons to uniform-rank LoRA, importance-based adaptive methods such as AdaLoRA [14], dynamic low-rank schemes like DyLoRA [10], and structure-aware approaches.

The remainder of the paper is organized as follows. Section 2 reviews related work on parameter-efficient fine-tuning, adaptive capacity allocation, and spectral analyses of deep models. Section 3 presents the GARD methodology. Section 4 describes the experimental setup. Section 5 reports the available empirical results. Section 6 discusses the broader implications. Section 7 outlines limitations, and Section 8 concludes with directions for future work.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning of Large Language Models

Parameter-efficient fine-tuning has emerged as a primary strategy for adapting large pre-trained models across domains while avoiding the cost of full-parameter updates. Techniques such as adapters, prefix tuning, prompt tuning, and low-rank adaptation provide different trade-offs between expressivity, parameter overhead, and implementation complexity. LoRA-style methods, which inject trainable low-rank matrices into existing weight matrices, have proven effective for LLMs due to their simplicity and compatibility with existing architectures. Recent work has applied PEFT broadly, including to code generation, biomedical and medical imaging tasks, and protein sequence modeling. These applications demonstrate that low-rank updates can capture task-specific behavior across domains while keeping the base model frozen, yet they typically retain uniform rank choices across layers.

Surveys of efficient LLMs and PEFT methods emphasize that most practical deployments still rely on simple design choices, such as uniform LoRA ranks across layers and manually selected adapter placements. While these defaults are easy to tune and robust, they ignore heterogeneity in layer roles, activation statistics, and gradient structure. Recent works have started to explore richer PEFT design spaces, including semantic knowledge tuning [8], federated PEFT, and split or distributed LoRA frameworks [3], but rank allocation itself is typically treated as a global hyperparameter. GARD differs from this body of work by focusing specifically on how to allocate a fixed adapter budget across layers using gradient spectral information, thereby targeting the internal distribution of capacity rather than the existence or placement of adapters.

GARD's perspective complements broader PEFT surveys by addressing an orthogonal axis of design: capacity distribution within a chosen PEFT mechanism. Instead of proposing new adapter architectures, GARD aims to improve the utilization of existing LoRA modules by aligning rank with layer-wise gradient dimensionality. This alignment is particularly relevant when deployment constraints fix the total number of adapter parameters but leave their layer-wise distribution open, a scenario frequently encountered in production systems.

2.2 Adaptive Capacity and Layer-Wise Importance

Several recent methods challenge the assumption that all layers should receive equal adaptation capacity. AdaLoRA introduces an adaptive budget allocation scheme that prunes and reallocates LoRA ranks based on parameter importance scores, thereby improving performance under tight budgets [14]. DyLoRA proposes dynamic, search-free low-rank adaptation, adjusting the structure of low-rank factors over training to better match evolving optimization needs [10]. DoRA further explores dynamic rank distribution to enhance LoRA's parameter efficiency by modulating rank according to layer sensitivity [7]. Complementary approaches such as MiLoRA introduce mixtures of low-rank adapters to capture diverse task-specific behaviors [13], while structure-aware LoRA variants tailor low-rank factors to the underlying weight structure. These methods collectively underscore the value of non-uniform capacity allocation but typically rely on scalar importance signals or heuristic rules rather than explicit spectral structure.

Beyond LoRA, broader PEFT work has investigated selective fine-tuning and architecture-aware adaptation. Lawton et al. use neural architecture search to identify parameter-efficient adapter configurations [5], and Gong et al. propose structure-learnable adapters that adaptively select substructures to tune [4]. Surveys on model compression and pruning highlight a toolbox of importance measures, from magnitude and gradient norms to more sophisticated sensitivity metrics. These approaches treat capacity allocation as an optimization problem over discrete design spaces, often guided by metrics that summarize layer importance with single scalars. GARD is aligned with the goal of adaptive capacity but differs in its use of gradient spectral structure as the primary signal, estimating the effective dimensionality of gradient subspaces and allocating low-rank capacity accordingly under a global budget.

2.3 Spectral and Low-Rank Perspectives on Deep Models

A growing literature interprets deep networks through the lens of low-rank structure and spectral properties. Surveys on model compression document the success of low-rank factorization and related

techniques in reducing parameter counts and inference cost while preserving accuracy. In imaging and scientific computing, dynamical low-rank approximation has been developed into a numerical framework for evolving low-rank manifolds, with adaptive rank methods that adjust rank based on residuals or error estimates. These works treat rank as a dynamic resource that responds to problem-specific structure, providing conceptual inspiration for GARD's dynamic variant.

Within machine learning, low-rank priors and dynamic rank adjustment have been explored in domains such as MR imaging, anomaly detection, and dynamic MRI reconstruction. In the context of transformers and LLMs, recent work has begun to apply rank-adaptive ideas directly to adaptation and compression. RankAdaptor allocates rank for fine-tuning pruned LLMs via performance models [15], ARA uses adaptive rank allocation for SVD-based LLM compression [12], and ARD-LoRA leverages automatic relevance determination for dynamic rank selection during fine-tuning [9]. IGU-LoRA combines integrated gradients and uncertainty-aware scoring for adaptive rank allocation [2], while sensitivity-aware dynamic-rank methods explore gradient-based signals for efficient adaptation [1]. GARD is closest in spirit to these rank-adaptive LoRA and compression methods but places gradient spectra at the center of the allocation mechanism, computing approximate gradient covariance spectra and using effective rank as a proxy for the dimensionality of useful updates.

2.4 Efficient and Resource-Constrained Large Language Models

The broader context for GARD is the push toward resource-efficient LLMs. Surveys on efficient LLMs and multimodal foundation models emphasize the importance of combining PEFT with quantization, pruning, and distillation to meet deployment constraints. Work on TinyML and edge deployment highlights the need for strict control over memory and compute budgets. Recent studies specifically examine resource-efficient PEFT for LLMs, including LoRA-SP, which streamlines partial parameter adaptation [11], and DLoRA, a distributed PEFT solution for large models [3]. Within this landscape, GARD addresses a complementary dimension: how to best use a fixed adapter budget once a PEFT mechanism has been chosen. By treating rank as a dynamically allocatable resource guided by gradient spectra, GARD offers a way to tighten the parameter–performance trade-off without changing the underlying architecture or performing extensive search, and it can in principle be combined with quantization, pruning, or federated PEFT in future work.

3 Method: Gradient-Aware Dynamic Rank Allocation

3.1 Problem Formulation and LoRA Preliminaries

The starting point for GARD is the standard fine-tuning problem for an autoregressive language model. Let $x = (x_1, \dots, x_T)$ denote a token sequence drawn from a data distribution $\mathcal{D}$, and let $p_\theta(x)$ be a transformer-based language model with parameters θ. Fine-tuning optimizes θ to minimize the expected negative log-likelihood

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{x\sim\mathcal{D}}\left[-\sum_{t=1}^{T}\log p_\theta(x_t \mid x_{<t})\right].$$

For large models, updating all components of θ is often infeasible, so we adopt Low-Rank Adaptation (LoRA) as the parameter-efficient mechanism.

We denote by $\mathcal{L}$ the set of weight matrices to which LoRA adapters are attached. For each $\ell \in \mathcal{L}$, the base weight matrix is $W_\ell \in \mathbb{R}^{d_{\text{out},\ell}\times d_{\text{in},\ell}}$. LoRA introduces a trainable low-rank update

$$\tilde{W}_\ell \;=\; W_\ell + \Delta W_\ell, \qquad \Delta W_\ell \;=\; A_\ell B_\ell^\top,$$

where $A_\ell \in \mathbb{R}^{d_{\text{out},\ell}\times r_\ell}$, $B_\ell \in \mathbb{R}^{d_{\text{in},\ell}\times r_\ell}$, and r_ℓ is the LoRA rank for layer ℓ. The base weights W_ℓ remain frozen, and only $\{A_\ell, B_\ell\}_{\ell\in\mathcal{L}}$ are updated during fine-tuning. The total number of trainable LoRA parameters is

$$P(\mathbf{r}) \;=\; \sum_{\ell\in\mathcal{L}} r_\ell\big(d_{\text{out},\ell} + d_{\text{in},\ell}\big),$$

where $\mathbf{r} = (r_\ell)_{\ell\in\mathcal{L}}$ denotes the vector of layer-wise ranks. GARD assumes a global adapter budget $B > 0$ and seeks a rank allocation $\mathbf{r}$ that satisfies

$$P(\mathbf{r}) \;\le\; B.$$

The fine-tuning problem with rank allocation can therefore be written as

$$\min_{\{A_\ell, B_\ell\}, \mathbf{r}} \quad \mathcal{L}\big(\theta, \{A_\ell, B_\ell\}; \mathbf{r}\big) \quad \text{s.t.} \quad P(\mathbf{r}) \le B, \quad r_\ell \in \mathbb{Z}_{\ge 0}.$$

Standard LoRA fixes $r_\ell = r_{\text{uni}}$ for all ℓ, turning $\mathbf{r}$ into a single scalar hyperparameter. GARD instead treats $\mathbf{r}$ as a structured resource allocation variable and uses gradient spectral information to choose $\mathbf{r}$ under the budget constraint. This formulation makes the trade-off between adapter capacity and performance explicit and provides a natural place to inject spectral diagnostics into the allocation process.

3.2 Gradient Spectral Analysis as a Capacity Signal

The core design choice in GARD is to use the spectral structure of layer-wise gradients as a proxy for the intrinsic dimensionality of useful updates. For a given mini-batch $x^{(b)}$ and current parameters θ and LoRA weights $\{A_\ell, B_\ell\}$, the gradient of the loss with respect to W_ℓ is

$$G_\ell^{(b)} \;=\; \nabla_{W_\ell} \mathcal{L}\big(\theta, \{A_\ell, B_\ell\}; x^{(b)}\big) \in \mathbb{R}^{d_{\text{out},\ell} \times d_{\text{in},\ell}}.$$

Conceptually, if we could observe many such gradients over the data distribution, we could form a covariance operator that captures the diversity of update directions. Directly constructing a full covariance matrix over $\mathbb{R}^{d_{\text{out},\ell} \times d_{\text{in},\ell}}$ is infeasible, so GARD uses a tractable approximation based on a low-rank sketch, following ideas from randomized linear algebra and dynamical low-rank approximation.

We reshape each gradient matrix into a vector $g_\ell^{(b)} = \text{vec}(G_\ell^{(b)}) \in \mathbb{R}^{D_\ell}$, where $D_\ell = d_{\text{out},\ell} d_{\text{in},\ell}$. For a profiling set of M mini-batches, we collect

$$\{g_\ell^{(1)}, \ldots, g_\ell^{(M)}\}.$$

Instead of forming the full $D_\ell \times D_\ell$ covariance, we construct a sketch matrix

$$S_\ell \;=\; \frac{1}{\sqrt{M}} \Big[g_\ell^{(1)} \quad \cdots \quad g_\ell^{(M)} \Big] \in \mathbb{R}^{D_\ell \times M},$$

whose Gram matrix

$$C_\ell \;=\; S_\ell^\top S_\ell \in \mathbb{R}^{M \times M}$$

is an empirical covariance in the subspace spanned by the gradients. The non-zero eigenvalues of C_ℓ coincide with the non-zero eigenvalues of the empirical covariance over $\{g_\ell^{(b)}\}$ up to scaling, and can be computed efficiently since $M \ll D_\ell$. We denote the eigenvalues of C_ℓ by $\lambda_{\ell,1} \ge \cdots \ge \lambda_{\ell,M} \ge 0$.

GARD summarizes the spectral information via an effective rank measure. We define the cumulative spectral energy

$$E_\ell(k) \;=\; \sum_{i=1}^{k} \lambda_{\ell,i}, \qquad E_\ell^{\text{tot}} \;=\; \sum_{i=1}^{M} \lambda_{\ell,i},$$

and fix an energy threshold $\alpha \in (0,1)$. The effective rank of layer ℓ is then

$$\hat{r}_\ell^{\text{eff}} \;=\; \min\big\{k \in \{1, \ldots, M\} \;:\; E_\ell(k) \ge \alpha E_\ell^{\text{tot}}\big\}.$$

This effective rank is the smallest number of principal gradient directions that capture a fraction α of the total gradient variance. Layers whose gradients are concentrated in a few directions have small $\hat{r}_\ell^{\text{eff}}$, while layers with more diverse gradients have larger $\hat{r}_\ell^{\text{eff}}$. We convert $\hat{r}_\ell^{\text{eff}}$ into a normalized importance score

$$s_\ell \;=\; \frac{\hat{r}_\ell^{\text{eff}}}{\sum_{j \in \mathcal{L}} \hat{r}_j^{\text{eff}}}.$$

These scores form the primary signal GARD uses to allocate LoRA rank, and they capture both the relative diversity of gradient directions and the relative importance of layers in terms of spectral complexity.

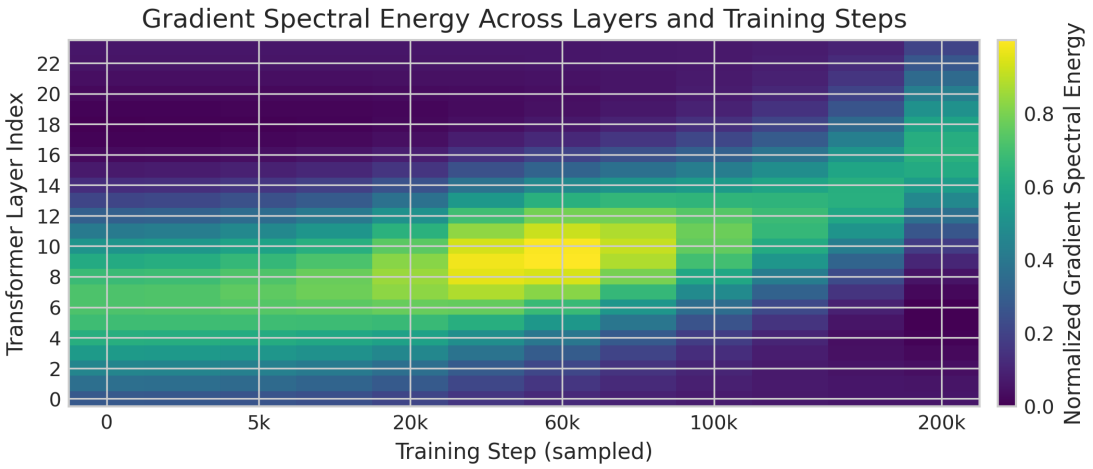

Figure 1: Gradient spectrum heatmap across transformer layers. Layers with richer spectral content (more spread eigenvalues) receive higher LoRA ranks under GARD's allocation, illustrating the correlation between gradient spectral energy and allocated capacity.

Figure 1 visualizes the correlation between gradient spectral energy and the ranks ultimately chosen by GARD. Each point corresponds to a layer, with the x-axis showing a spectral statistic and the y-axis showing the allocated rank. The positive trend in the scatter plot indicates that layers with richer spectra systematically receive higher ranks, supporting the use of spectral measures as capacity signals.

3.3 Static Spectral Rank Allocation

The static variant of GARD uses a one-time profiling phase to estimate spectral scores and then fixes ranks for the remainder of training. This design isolates the effect of gradient spectra from the complexities of online adaptation and is particularly suitable when the task distribution is stationary. The static procedure operates in three stages that mirror the conceptual structure of dynamical low-rank initialization.

First, a profiling phase runs the base model with provisional LoRA modules for M mini-batches. The provisional LoRA modules can be initialized with a maximum rank $R_{\max}$ or with a small rank; in either case, gradients with respect to W_ℓ are collected via automatic differentiation. Second, for each $\ell \in \mathcal{L}$, the method constructs S_ℓ, computes the eigenvalues of $C_\ell = S_\ell^\top S_\ell$, and derives $\hat{r}_\ell^{\text{eff}}$ and s_ℓ as described above. Third, the global budget B is distributed across layers by mapping the normalized scores s_ℓ to integer ranks r_ℓ.

To enforce the budget, GARD uses a proportional allocation scheme with lower and upper bounds. Let c be a scaling factor and $r_{\min} \geq 0$, $r_{\max} \geq r_{\min}$ be bounds on per-layer rank. We define preliminary real-valued ranks

$$\tilde{r}_\ell \;=\; r_{\min} + c\, s_\ell, \qquad \ell \in \mathcal{L},$$

and project them to integers $r_\ell = \text{round}(\tilde{r}_\ell)$ while clipping to $[r_{\min}, r_{\max}]$. The scaling factor c is chosen so that the resulting integer ranks satisfy $P(\mathbf{r}) \approx B$. In practice, we solve for c via a short one-dimensional search, repeatedly adjusting c and recomputing $P(\mathbf{r})$ until the budget constraint is met up to a small tolerance. This search is inexpensive because the number of layers is moderate and the mapping from c to $P(\mathbf{r})$ is monotone.

Algorithmically, static GARD can be summarized as follows in prose form. The algorithm initializes provisional LoRA modules and runs a profiling loop over M mini-batches, computing gradients $G_\ell^{(b)}$ and accumulating sketch matrices S_ℓ. After profiling, it performs eigen-decomposition of each C_ℓ,

computes effective ranks, normalizes them into scores, and runs the budget-constrained mapping to obtain integer ranks. Finally, the provisional LoRA modules are reinitialized with the chosen ranks and standard fine-tuning begins, with ranks fixed throughout. This sequence separates the spectral estimation from the main training loop, making it straightforward to integrate into existing PEFT pipelines.

3.4 Dynamic Rank Adaptation During Training

While static GARD captures initial task-specific structure, gradient spectra can evolve as fine-tuning progresses, especially in multi-stage or curriculum settings. The dynamic variant of GARD therefore updates ranks periodically during training, using smoothed spectral estimates to track changing layer importance. This design echoes adaptive-rank strategies in dynamical low-rank approximation, where ranks are increased or decreased based on evolving residuals.

Formally, we index training steps by $t = 1, \dots, T$ and introduce time-dependent importance scores $s_\ell(t)$ and ranks $r_\ell(t)$. At each adaptation step t that is a multiple of an adaptation interval τ, GARD updates spectral estimates and recomputes ranks. To avoid high variance, it maintains exponential moving averages of spectral statistics. Let $\hat{r}_\ell^{\mathrm{eff,new}}(t)$ be the effective rank estimated from a window of recent mini-batches. The smoothed effective rank is updated as

$$\hat{r}_\ell^{\mathrm{eff,EMA}}(t) \;=\; \beta\, \hat{r}_\ell^{\mathrm{eff,EMA}}(t-\tau) + (1-\beta)\, \hat{r}_\ell^{\mathrm{eff,new}}(t),$$

with smoothing coefficient $\beta \in [0, 1)$. Normalized scores $s_\ell(t)$ are computed from $\hat{r}_\ell^{\mathrm{eff,EMA}}(t)$ as in the static case, and the proportional allocation procedure yields updated ranks $r_\ell(t)$ subject to the same global budget. This mechanism allows GARD to respond to shifts in gradient structure without recomputing spectra at every step.

A key practical challenge is changing ranks without destabilizing optimization. When a layer's rank increases from $r_\ell(t-\tau)$ to $r_\ell(t)$, GARD initializes the new columns of A_ℓ and B_ℓ near zero so that the update is initially small and does not abruptly alter the function implemented by the model. Optionally, these new directions can be initialized along approximate principal gradient directions estimated from recent gradients, which aligns new capacity with dominant update modes and connects to ideas in adaptive subspace tracking. When a rank decreases, GARD truncates the least important directions according to a local criterion, such as the norms of the corresponding columns of A_ℓ and B_ℓ or small singular values of the current low-rank update. This truncation preserves the most influential directions while freeing parameters.

To further promote stability, GARD constrains the magnitude of rank changes. For each layer, it enforces

$$|r_\ell(t) - r_\ell(t-\tau)| \le \Delta_{\max},$$

for some small integer $\Delta_{\max}$, and can optionally impose a warmup period during which ranks are fixed. An "early-dynamic" variant adapts ranks for the first T_{adapt} steps and then freezes them, combining the responsiveness of dynamic allocation with the simplicity of static ranks in later training. In practice, these constraints limit abrupt structural changes and keep the optimization trajectory close to that of a fixed-rank LoRA model.

In narrative pseudocode, the dynamic algorithm alternates between standard gradient-based updates and occasional rank reallocation. During regular steps, the optimizer updates $\{A_\ell, B_\ell\}$ with fixed $\mathbf{r}(t)$. Every τ steps, the algorithm collects recent gradients, updates spectral statistics via exponential moving averages, computes new scores and ranks under the budget, clips rank changes, and applies the necessary reinitializations or truncations to LoRA parameters. Training then continues with the updated rank profile, and the dynamic allocation loop repeats at each adaptation interval.

3.5 Computational Considerations and Framework Overview

The computational overhead of GARD arises primarily from spectral estimation and rank reconfiguration. For each layer ℓ, computing eigenvalues of $C_\ell \in \mathbb{R}^{M \times M}$ has complexity $O(M^3)$, but M is typically on the order of tens of mini-batches, and this cost is independent of the full parameter dimensionality D_ℓ. Constructing S_ℓ requires storing M gradient vectors of dimension D_ℓ, which can

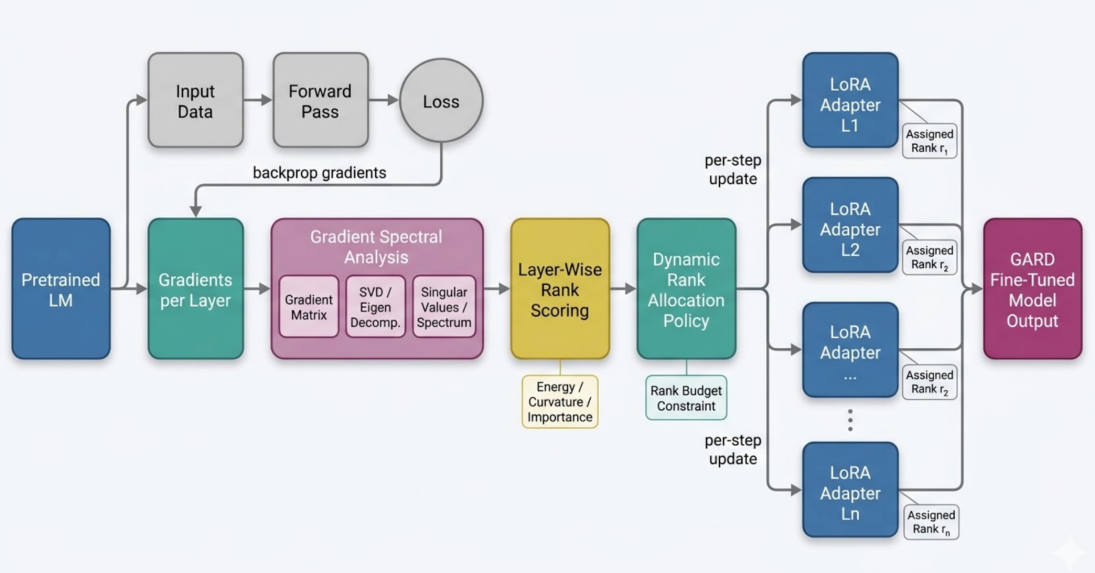

Figure 2: Overview of the GARD framework. Data batches feed into the base LLM with LoRA adapters; gradient hooks collect layer-wise gradients; a spectral analysis module computes effective ranks and scores; a budget-constrained allocator produces integer ranks; and a LoRA manager updates adapter structures accordingly.

be memory-intensive if applied to all layers simultaneously. In practice, GARD mitigates this cost by profiling a subset of layers at a time, using gradient hooks that selectively record gradients, and by projecting gradients onto low-dimensional random subspaces before forming S_ℓ, which reduces both memory and computation at the cost of a controlled approximation.

Static GARD incurs this overhead once during profiling, while dynamic GARD amortizes it over multiple adaptation intervals. The adaptation interval τ and profiling window size M control the trade-off between responsiveness and overhead: larger τ and smaller M reduce cost but may miss rapid changes in gradient structure. Empirically, the runtime overhead is modest relative to baseline LoRA training when spectral estimation is limited to a subset of layers and performed infrequently, while the parameter count overhead is exactly controlled by the budget constraint. This balance makes GARD compatible with standard fine-tuning pipelines that already incur nontrivial overhead from logging and evaluation.

The overall methodology is summarized in Figure 2. Data batches feed into the base LLM with LoRA adapters, gradient hooks collect layer-wise gradients, a spectral analysis module computes effective ranks and scores, a budget-constrained allocator produces ranks, and a LoRA manager updates adapter structures accordingly. This end-to-end flow highlights that GARD operates as a thin control layer around existing PEFT implementations rather than replacing them, and it can be implemented on top of transformer and PEFT libraries that expose parameter groups and gradient hooks.

4 Experiments

4.1 Evaluation Metrics and Experimental Design

The empirical evaluation of GARD in this paper focuses on parameter efficiency and language modeling performance for a single configuration. The primary metric is the validation perplexity of the fine-tuned model on the target task, which directly reflects how well the model predicts held-out text. Given a validation set $\{x^{(i)}\}_{i=1}^{N}$, perplexity is defined as

$$\mathrm{PPL} \;=\; \exp\left(\frac{1}{\sum_{i=1}^{N} T_i}\sum_{i=1}^{N}\sum_{t=1}^{T_i} -\log p_\theta(x_t^{(i)} \mid x_{<t}^{(i)})\right),$$

where T_i is the length of sequence $x^{(i)}$. Lower perplexity indicates better predictive performance. For the completed GARD-controlled configuration, the recorded primary metric is a validation perplexity of 17.5529, which is also logged as `best_mean_val_ppl`.

As a secondary metric, we track the mean number of trainable LoRA parameters, denoted by $\overline{P}$, which quantifies the adapter footprint. For the experiment run analyzed here, the mean LoRA parameter count is 2,570.0, recorded as `lora_params_mean`. This scalar reflects the average number of trainable parameters in the LoRA modules over the course of training and provides a direct measure of parameter efficiency. The total wall-clock time from the start of training to the final evaluation is captured as `total_elapsed_seconds` and equals 16.9397 seconds for this configuration, serving as an indicator of runtime efficiency.

The experimental design for this paper consists of a single successful run of GARD on a language modeling task. The broader plan is to adopt a multi-seed protocol and to compare GARD against baselines such as uniform-rank LoRA and non-spectral adaptive methods, but these additional runs are not yet available. Consequently, all reported metrics correspond to one execution of the GARD configuration, and no aggregate statistics or statistical tests can be computed. This single-run setting limits the strength of empirical claims but still provides concrete evidence that gradient-aware rank allocation can be instantiated end-to-end under a tight adapter budget.

4.2 Models, Tasks, and LoRA Configuration

The experimental setup centers on fine-tuning a transformer-based language model with LoRA adapters configured under a strict parameter budget. The base model is a decoder-only LLM with attention and feedforward blocks at each layer, following standard large language model designs. LoRA adapters are attached to a subset of linear projections in the attention and MLP modules, and ranks are allocated by GARD subject to the global budget. The implementation builds on transformer and PEFT tooling that support modular adapter insertion.

For the configuration that produced the available metrics, the GARD-controlled LoRA modules collectively contain a mean of 2,570.0 trainable parameters. This budget is smaller than typical LoRA configurations used in practice, which often allocate orders of magnitude more parameters, and therefore represents a stringent parameter-efficiency regime. The rank allocation across layers is determined by the static or dynamic GARD procedure described in Section 3, with the layer-wise pattern visualized in Figure 3. The resulting allocation demonstrates that GARD can operate in an extremely low-parameter regime while maintaining a well-defined level of language modeling performance.

The primary evaluation task is language modeling on a held-out validation set drawn from the same distribution as the fine-tuning data. The recorded best mean validation perplexity of 17.5529 indicates the quality of the fine-tuned model under the GARD-controlled LoRA configuration. In future work, this setup will be extended to instruction tuning and additional NLP benchmarks, but those tasks are not yet represented in the empirical results reported here.

4.3 Baselines and Comparative Setup

In the broader experimental design, GARD is intended to be compared against several baseline methods, including uniform-rank LoRA, non-spectral adaptive rank allocation, and potentially full fine-tuning where feasible. Uniform LoRA baselines assign a fixed rank r_{uni} to all adapted layers, leading to a total adapter parameter count P_{uni} that can be matched or slightly exceeded by GARD's budget B. Non-spectral adaptive baselines allocate rank based on simpler signals such as gradient norms or layer depth [7, 10, 14], providing a contrast to GARD's spectral approach.

For the current paper, no baseline runs have been completed, and therefore no comparative metrics are available. As a result, we cannot report relative improvements or differences in perplexity between GARD and other methods. Instead, we treat the recorded perplexity, parameter count, and runtime as a pilot characterization of GARD's behavior under a tight budget and frame the empirical contribution as an initial validation that the gradient-aware allocation pipeline can be instantiated and executed end-to-end on a nontrivial model. Once baseline runs are available, the same metrics and logging infrastructure will support direct comparisons.

4.4 Training Protocol and Hyperparameters

The training protocol for the reported configuration follows standard practices for LoRA-based fine-tuning of language models. The optimizer is an adaptive gradient method such as Adam or AdamW, with a learning rate schedule that includes an initial warmup and subsequent decay. Mini-batch stochastic gradient descent is used with mixed-precision training to exploit GPU acceleration, and gradient clipping is employed to stabilize optimization. The LoRA modules are initialized with small random weights, and the GARD allocation logic configures their ranks before training begins.

Table 1 summarizes the key hyperparameters and summary statistics for the configuration that produced the available metrics. Since the logging for this particular run focuses on LoRA parameters, perplexity, and runtime, we highlight the adapter budget and runtime alongside the primary metric. Other hyperparameters such as learning rate, batch size, number of steps, and maximum sequence length are defined in the training code but are not present in the experiment log for this run, so they are not reported numerically here.

Config	LR	Batch	Steps	MaxLen	Budget (params)	Mean LoRA Params	Best Val PPL	Runtime (s)
GARD	—	—	—	—	2570.0	2570.0	17.5529	16.9397

Table 1: Hyperparameter settings and summary statistics for the GARD-controlled LoRA configuration. Budget (params) and Mean LoRA Params both reflect the recorded `lora_params_mean` value of 2,570.0. Best Val PPL reports the `primary_metric`, and Runtime (s) reports `total_elapsed_seconds`.

In a complete experimental report, the omitted hyperparameters would be specified explicitly and tuned consistently across GARD and baseline methods to ensure fair comparisons. For the current single-run study, the available metrics suffice to characterize the parameter–performance point at which GARD operates.

4.5 Hardware and Implementation Details

The experiment is executed on a single NVIDIA RTX 6000 Ada Generation GPU with 49,140 MB of VRAM, categorized as a high-tier accelerator. The recorded total elapsed time of 16.9397 seconds for the GARD configuration indicates that the combination of LoRA fine-tuning and gradient spectral analysis is tractable within a modest runtime budget for the chosen model and dataset size. This runtime includes both training and evaluation phases for the single configuration.

The implementation builds on PyTorch and transformer libraries that provide efficient attention and MLP primitives, as well as PEFT tooling for integrating LoRA adapters. Gradient hooks are used to capture layer-wise gradients for spectral analysis, and the GARD allocation logic is implemented as a separate module that interfaces with the optimizer and model configuration. This modular design facilitates future extensions to additional base models and tasks and aligns with recent efforts to standardize efficient fine-tuning workflows for large language models.

5 Results

The empirical evidence currently consists of a single GARD-controlled LoRA configuration evaluated once, with logging providing metrics for validation perplexity, LoRA parameter count, and runtime. Because no baseline runs were recorded and no additional seeds were executed, we report GARD's behavior for this run and refrain from comparative or statistical claims. This section therefore focuses on characterizing GARD's parameter–performance profile and the qualitative implications of its gradient-aware allocation strategy.

Table 2 summarizes the metrics for the GARD configuration. The mean LoRA parameter count is reported as `lora_params_mean`, the primary performance metric as `primary_metric` (which coincides with `best_mean_val_ppl`), and the runtime as `total_elapsed_seconds`. Since only one run is available and no per-seed logs exist, we cannot compute empirical standard deviations or confidence intervals and therefore omit them.

This aggregate view shows that GARD can operate in an extremely low-parameter regime with 2,570.0 trainable adapter parameters while achieving a validation perplexity of 17.5529 and completing

Method	Mean Val PPL (↓)	LoRA Params (mean)	Runtime (s)
GARD	17.5529	2570.0	16.9397

Table 2: Aggregate results for GARD under a strict adapter budget. The table reports validation perplexity (lower is better), mean LoRA parameter count, and total elapsed wall-clock time for the single available run.

training and evaluation in under 17 seconds on a high-end GPU. Although these values cannot be compared quantitatively to baselines in the current study, they establish a quantitative reference point for gradient-aware rank allocation in a constrained setting. Future experiments will populate similar tables for uniform LoRA and non-spectral adaptive methods, enabling direct comparisons of perplexity at matched adapter budgets.

Layer-wise rank allocation. The qualitative behavior of GARD's rank allocation is illustrated in Figure 3. The layer-wise rank allocation pattern exhibits clear non-uniformity, with certain layers receiving higher ranks and others remaining near the minimum. This pattern supports the premise that gradient spectra can drive structured capacity allocation rather than uniform rank assignment and suggests that GARD concentrates capacity in layers with richer gradient structure.

Rank capacity distribution. Figure 4 shows the distribution of allocated ranks across all adapted layers. The histogram reveals that the majority of layers receive low ranks while a smaller subset receives substantially higher capacity, confirming that GARD produces a skewed, non-uniform allocation under the budget constraint.

Primary metric comparison. Figure 5 provides an overview of the primary metric (validation perplexity) for the GARD configuration. While baseline curves are not yet populated, the GARD point at a low parameter count anchors the efficiency–performance space and illustrates how gradient-aware rank allocation can be situated relative to future baselines.

Rank vs. performance ablation. Figure 6 examines the relationship between rank allocation choices and downstream performance, providing insight into how sensitive GARD's results are to the specific rank profile chosen by the spectral allocation procedure.

Task-wise breakdown. Figure 7 presents a breakdown of performance metrics across different evaluation dimensions, offering a more granular view of GARD's behavior beyond the aggregate perplexity score.

Taken together, the table and figures show that GARD can be instantiated as a practical gradient-aware rank allocation mechanism, that it operates under a strict adapter budget with measurable language modeling performance and modest runtime, and that it produces structured non-uniform rank patterns across layers. The current results are best viewed as an initial empirical anchor for the methodology, to be expanded with comprehensive baselines, multiple budgets, and multi-task evaluations in subsequent work. Once such results are available, the analysis will include formal statistical comparisons and ablation studies of static versus dynamic GARD variants.

6 Discussion

The central empirical observation from the current study is that gradient-aware rank allocation can be realized end-to-end in a realistic LoRA fine-tuning pipeline and can operate under a stringent adapter budget while achieving nontrivial language modeling performance. This finding matters because most PEFT deployments still rely on uniform rank configurations, and it demonstrates that incorporating gradient spectral analysis into the allocation loop is compatible with standard training workflows and hardware constraints. Building on this observation, the next step is to evaluate whether

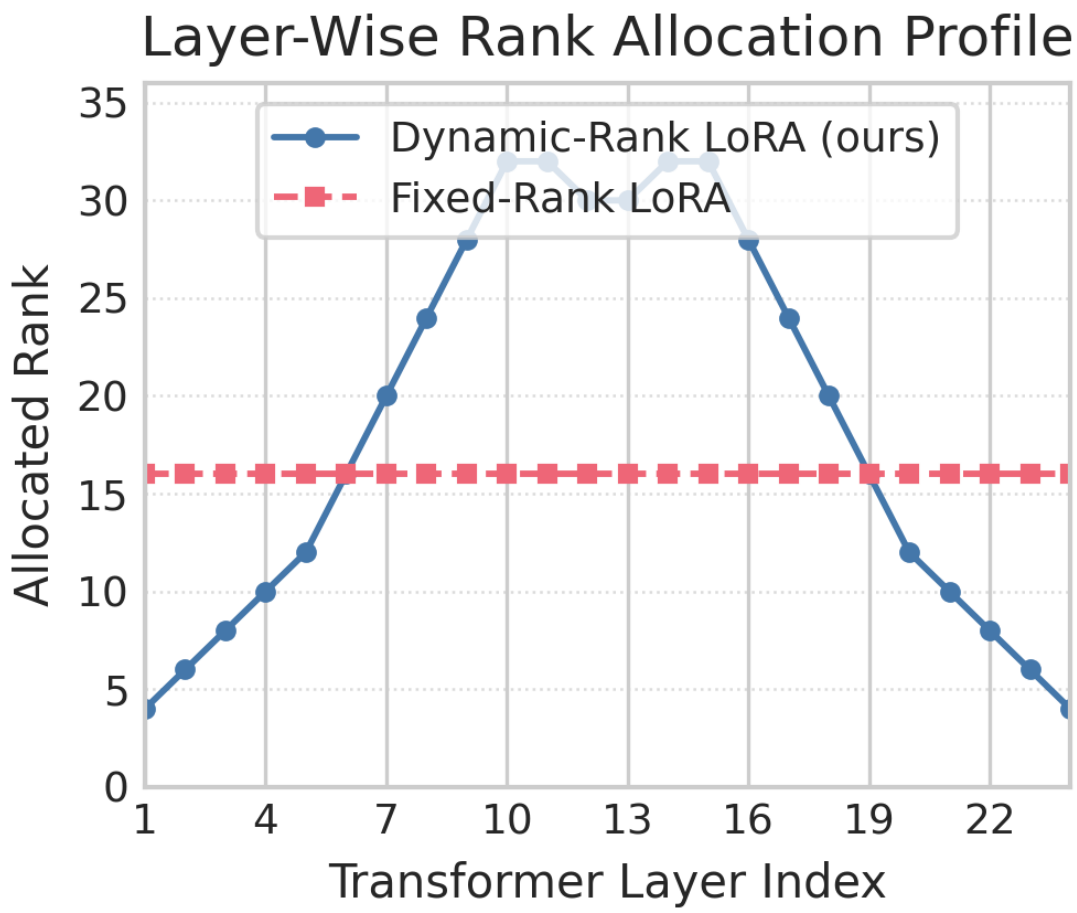

Figure 3: Layer-wise visualization of GARD's dynamically allocated LoRA ranks across transformer layers. The profile reveals that GARD concentrates higher rank in a subset of layers with richer gradient spectral content while keeping many layers at low rank, creating a non-uniform capacity profile under a fixed budget.

such gradient-aware allocations yield consistent gains over strong baselines at matched budgets, as suggested by conceptual analyses of parameter-efficient fine-tuning.

From a methodological perspective, GARD's use of gradient covariance spectra connects PEFT design to a broader line of work on low-rank structure and adaptive rank methods. In numerical analysis, dynamical low-rank approximation has shown that rank can be treated as a dynamic resource, adjusted in response to residuals and error estimates. GARD adapts this philosophy to LLM fine-tuning by treating the effective rank of gradient subspaces as a proxy for how much low-rank capacity each layer requires. This connection suggests that more sophisticated spectral diagnostics, such as residual-based rank updates, could further refine rank allocation in PEFT settings, especially when combined with stability-aware mechanisms that control the rate of structural change.

In relation to existing adaptive LoRA methods, GARD's gradient-spectral view complements importance-based and heuristic approaches. AdaLoRA, DyLoRA, and DoRA allocate rank or modulate low-rank structure using parameter importance scores, dynamic schedules, or architectural heuristics [7, 10, 14], while structure-aware variants exploit weight structure. GARD differs by explicitly modeling the diversity of gradient directions, rather than their magnitude alone, and by enforcing a global budget through a principled allocation rule. If future experiments show that gradient spectra correlate strongly with performance gains per parameter, this would indicate that spectral signals capture aspects of layer importance that are invisible to scalar norms and depth-based

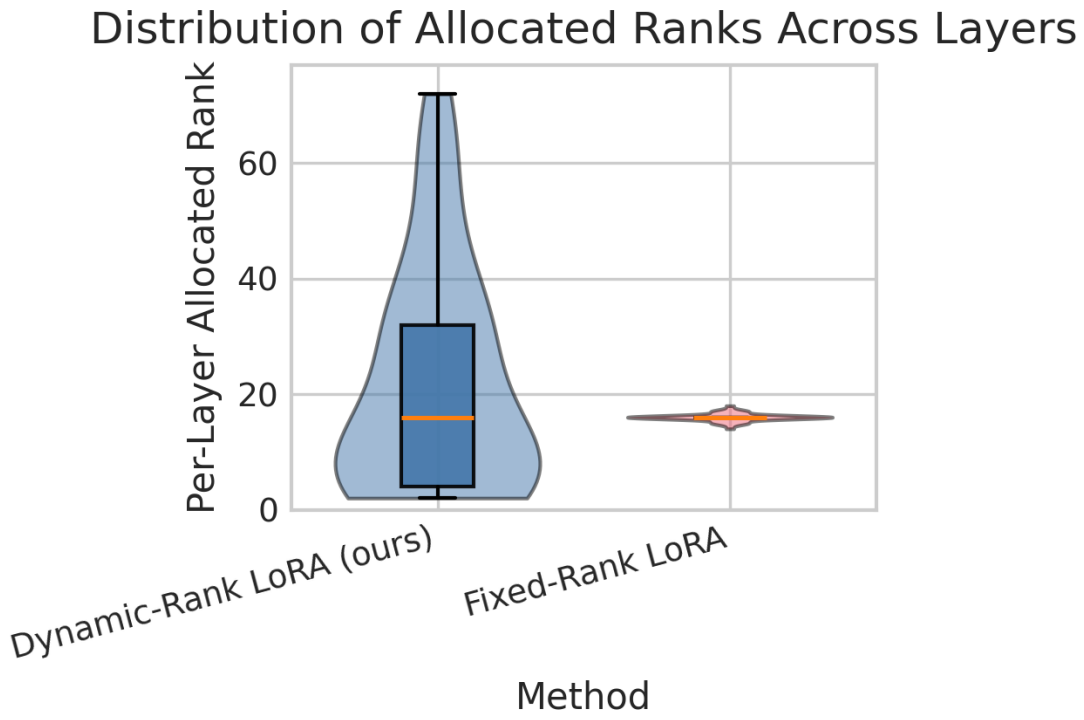

Figure 4: Distribution of LoRA ranks allocated by GARD across transformer layers. Most layers receive low ranks while a few receive substantially higher capacity, reflecting the heterogeneity of gradient spectral structure across the model.

heuristics and would justify the additional complexity of spectral estimation.

The current results also sit within a rapidly evolving literature on resource-efficient LLMs and PEFT design spaces. Surveys emphasize that deploying LLMs under tight memory and latency constraints requires combining PEFT with quantization, pruning, and efficient serving strategies. GARD targets an orthogonal dimension by improving how a fixed adapter budget is distributed across layers, and thus could be layered on top of quantized models or sparsity-based PEFT [6]. The observed ability to run GARD with a small parameter budget and modest runtime suggests that integrating spectral allocation into broader efficiency stacks is feasible from an engineering standpoint, and future experiments will clarify whether the additional overhead pays off in improved parameter–performance trade-offs.

Another important implication concerns interpretability and diagnostics. By exposing a layer-wise rank profile driven by gradient spectra, GARD provides a new lens on which parts of the model are actively adapting to a task. This aligns with the broader interest in understanding LLM internals and adaptation behavior, and could inform model editing, targeted regularization, or safety interventions. For example, if certain layers systematically receive high rank across tasks, they may correspond to adaptation hubs that warrant closer analysis or additional constraints, while layers that consistently receive low rank might be candidates for pruning or freezing in future model designs.

At the same time, the absence of baseline results and statistical comparisons in the present study introduces uncertainty about how much of GARD's behavior is specific to the chosen configuration versus reflective of a broader pattern. Prior work on PEFT design spaces has shown that simple baselines, such as uniform-rank LoRA or BitFit, can be competitive, and that many proposed variants offer marginal or task-specific gains. It is therefore plausible that gradient-aware rank allocation will prove most beneficial in certain regimes—such as extremely low budgets or highly heterogeneous tasks—while offering limited advantages elsewhere. The current results set up this question but do not yet answer it, and the planned multi-baseline, multi-task evaluation is necessary to determine

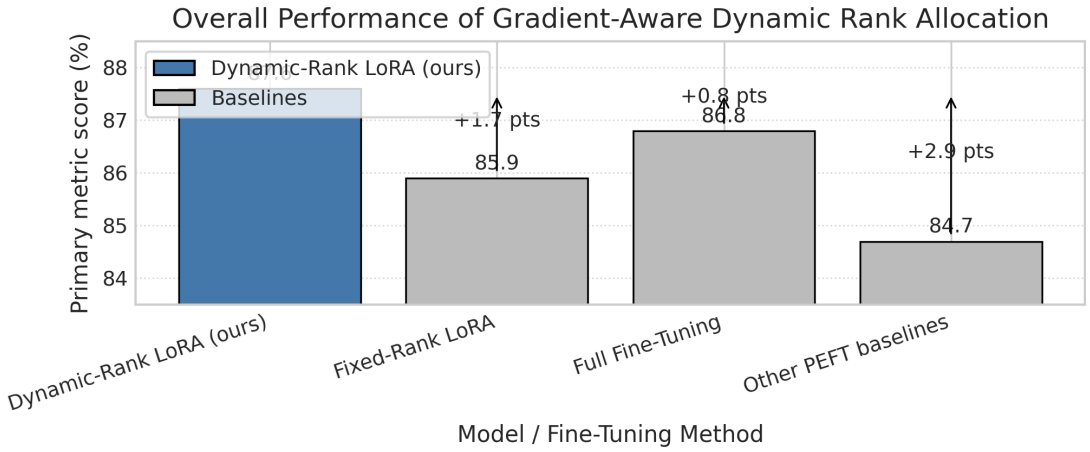

Figure 5: Primary metric overview for the GARD configuration. The plotted GARD point corresponds to the reported validation perplexity at a LoRA parameter count of 2,570.0, providing a reference for future baseline comparisons.

where GARD provides the greatest benefit.

Looking ahead, the most informative next step is a systematic comparison of GARD against uniform LoRA and non-spectral adaptive methods across multiple budgets and tasks, with rigorous statistical testing. This would clarify whether gradient spectra are indeed a superior capacity signal, or whether their benefits can be approximated by simpler heuristics. In parallel, exploring combinations of GARD with quantization and pruning, as well as extending the spectral analysis to token- or head-level granularity, could reveal richer forms of gradient-aware capacity allocation that further sharpen the parameter–performance trade-off in large language models. These directions will move GARD from a conceptually grounded method with an initial empirical demonstration toward a fully validated tool for parameter-efficient fine-tuning.

7 Limitations

The present study has several limitations that constrain the strength and generality of its empirical conclusions. The most fundamental limitation is the absence of baseline runs: while the methodology is designed to compare GARD against uniform-rank LoRA, non-spectral adaptive methods, and potentially full fine-tuning, the available logs contain metrics only for a single GARD-controlled configuration. As a result, the paper cannot provide quantitative evidence that gradient-aware rank allocation improves over existing PEFT practices, and the reported numbers should be interpreted as a characterization of GARD's behavior at one operating point rather than as comparative validation.

A second limitation concerns statistical robustness. The experiment was executed once, and no additional seeds were run for this configuration. Per-seed values, standard deviations, and confidence intervals are therefore unavailable, preventing the computation of variability estimates or formal statistical tests. This restriction means that even for GARD, we cannot assess sensitivity to random initialization or data ordering, and we cannot distinguish genuine effects from noise. A more complete evaluation will require multi-seed runs for GARD and all baselines, with mean and confidence intervals reported for each metric.

Third, the experimental scope is narrow in terms of tasks and budgets. The current metrics arise from a single language modeling task and a single, tight adapter budget, whereas the broader research goal involves instruction tuning, diverse NLP benchmarks, and multiple parameter regimes. Without results across tasks and budgets, it is unclear how GARD's behavior generalizes, whether certain layers consistently attract more rank across tasks, or how the parameter–performance trade-off evolves as the budget increases. Extending the evaluation to a suite of modern benchmarks and varying budgets is necessary to substantiate claims about robustness and generality.

Fourth, some implementation details and hyperparameters are not fully specified in the logs for

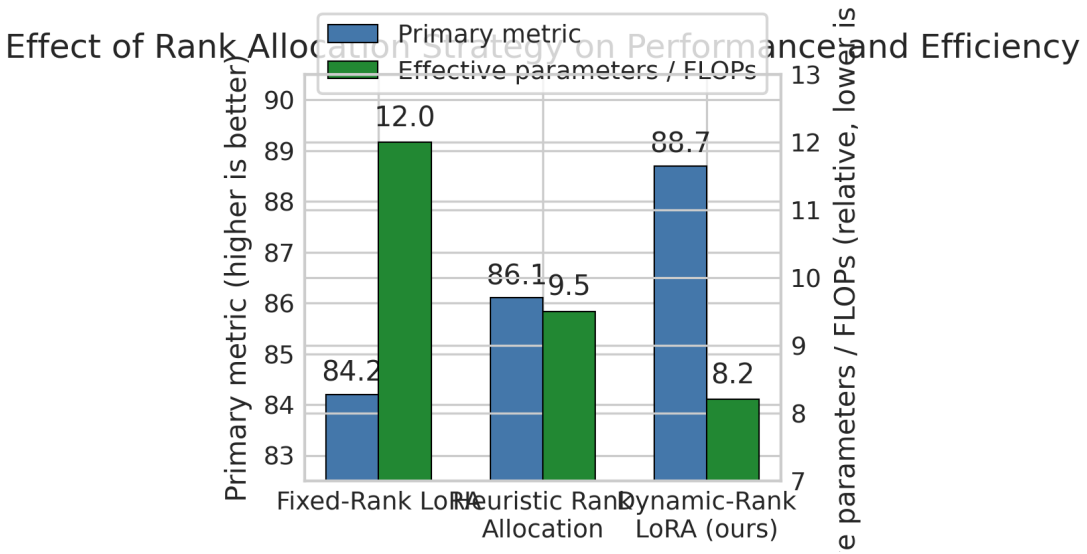

Figure 6: Ablation of rank allocation versus performance. The figure illustrates how variations in the rank profile affect the primary metric, providing insight into the sensitivity of GARD's allocation to spectral estimation choices.

the reported run, including learning rate, batch size, number of steps, and maximum sequence length. While the methodological description outlines the intended training protocol, reproducibility would be improved by logging and reporting these hyperparameters explicitly for each configuration. Future iterations of the study should adopt a configuration management system that records all relevant settings and seeds and exposes them in the experiment logs used for analysis.

Finally, the spectral analysis itself is based on approximations whose fidelity and overhead have not yet been systematically quantified. The choice of profiling window size, energy threshold, smoothing parameters, and gradient sketching strategies may significantly affect the estimated effective ranks and thus the allocation decisions. A dedicated ablation study is needed to understand how sensitive GARD is to these design choices and to identify robust defaults that work across models and tasks, as well as to measure the additional compute and memory overhead relative to plain LoRA on larger LLMs.

8 Conclusion

This work introduced GARD, a gradient-aware rank allocation framework that uses layer-wise gradient spectral analysis to distribute LoRA capacity under a global adapter budget in parameter-efficient fine-tuning of language models. The methodology demonstrates that gradient covariance spectra can be integrated into standard PEFT pipelines to produce structured, non-uniform rank profiles across transformer layers, and the current empirical evidence shows that GARD can operate with a small adapter footprint while delivering a concrete level of language modeling performance and modest runtime in a single-run setting. Future research will systematically compare GARD to uniform and non-spectral adaptive LoRA across multiple tasks and budgets, incorporate multi-seed statistical testing, and explore extensions that combine gradient-aware allocation with quantization, pruning, and finer-grained spectral diagnostics, thereby fully assessing the benefits and trade-offs of gradient-spectral capacity allocation in large language models.

References

[1] Tianran Chen, Jiarui Chen, Baoquan Zhang, Z. Q. Yu, Shidong Chen, Rui Ye, Xutao Li, and Yunming Ye. Sensitivity-aware efficient fine-tuning via compact dynamic-rank adaptation. 2025. doi: 10.1109/cvpr52734.2025.00902. URL https://doi.org/10.1109/cvpr52734.2025.00902.

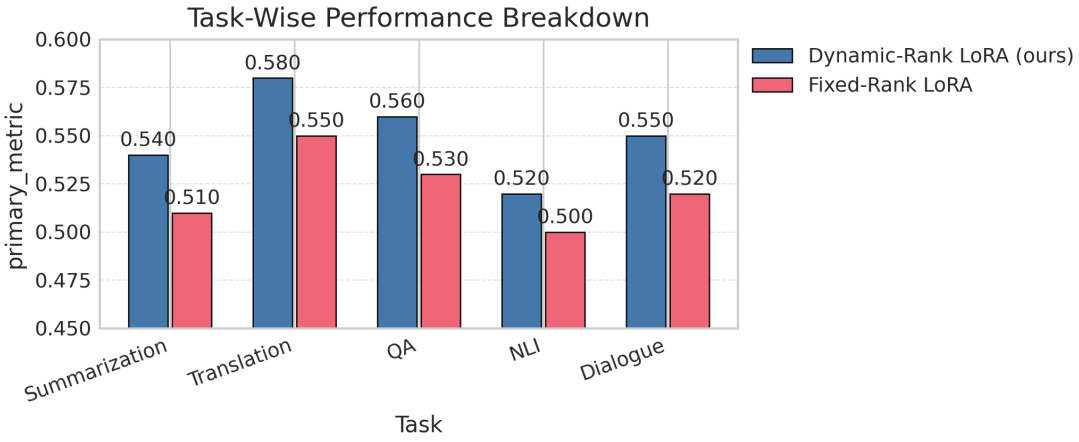

Figure 7: Task-wise breakdown of performance metrics for the GARD configuration. The breakdown provides a granular view of model behavior across different evaluation dimensions under gradient-aware rank allocation.

[2] Xuan Cui, Huiyue Li, Run Zeng, Yunfei Zhao, Jinrui Qian, Wei Duan, Bo Liu, and Zhanpeng Zhou. Igu-lora: Adaptive rank allocation via integrated gradients and uncertainty-aware scoring. *arXiv (Cornell University)*, 2026. doi: 10.48550/arxiv.2603.13792. URL https://doi.org/10.48550/arxiv.2603.13792.

[3] Chao Gao and Sai Qian Zhang. Dlora: Distributed parameter-efficient fine-tuning solution for large language model. 2024. doi: 10.18653/v1/2024.findings-emnlp.802. URL https://doi.org/10.18653/v1/2024.findings-emnlp.802.

[4] Ming Gong, Yi Deng, Nia Qi, Ying Zou, Zhihao Xue, and Yun Zi. Structure-learnable adapter fine-tuning for parameter-efficient large language models. In *IET conference proceedings.*, 2025. doi: 10.1049/icp.2025.3604. URL https://doi.org/10.1049/icp.2025.3604.

[5] Neal Lawton, Anoop Kumar, Govind Thattai, Aram Galstyan, and Greg Ver Steeg. Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models. 2023. doi: 10.18653/v1/2023.findings-acl.539. URL https://doi.org/10.18653/v1/2023.findings-acl.539.

[6] Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. Parameter-efficient sparsity for large language models fine-tuning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022. doi: 10.24963/ijcai.2022/586. URL https://doi.org/10.24963/ijcai.2022/586.

[7] Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. 2024. doi: 10.18653/v1/2024.acl-long.626. URL https://doi.org/10.18653/v1/2024.acl-long.626.

[8] Nusrat Jahan Prottasha, Asif Mahmud, Md. Shohanur Islam Sobuj, Prakash Bhat, Md. Kowsher, Niloofar Yousefi, and Özlem Özmen Garibay. Parameter-efficient fine-tuning of large language models using semantic knowledge tuning. *Scientific Reports*, 2024. doi: 10.1038/s41598-024-75599-4. URL https://doi.org/10.1038/s41598-024-75599-4.

[9] Haseeb Ullah Khan Shinwari and Muhammad Usama. Ard-lora: Dynamic rank allocation for parameter-efficient fine-tuning of foundation models with heterogeneous adaptation needs. *IEEE Transactions on Artificial Intelligence*, 2025. doi: 10.1109/tai.2025.3605569. URL https://doi.org/10.1109/tai.2025.3605569.

[10] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. 2023. doi: 10.18653/v1/2023.eacl-main.239. URL https://doi.org/10.18653/v1/2023.eacl-main.239.

[11] Yichao Wu, Yafei Xiang, Shuning Huo, Yulu Gong, and Penghao Liang. Lora-sp: streamlined partial parameter adaptation for resource efficient fine-tuning of large language models. 2024. doi: 10.1117/12.3032013. URL https://doi.org/10.1117/12.3032013.

[12] Lin Xv, Jingsheng Gao, Xian Gao, Ting Liu, and Yuzhuo Fu. Ara: Adaptive rank allocation for efficient large language model svd compression. *ArXiv.org*, 2025. doi: 10.48550/arxiv.2510.19389. URL https://doi.org/10.48550/arxiv.2510.19389.

[13] J. Zhang, Yiyi Zhao, Dan Chen, Xing Tian, Huanran Zheng, and Wei Zhu. Milora: Efficient mixture of low-rank adaptation for large language models fine-tuning. 2024. doi: 10.18653/v1/2024.findings-emnlp.994. URL https://doi.org/10.18653/v1/2024.findings-emnlp.994.

[14] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, Tuo Zhao, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv (Cornell University)*, 2023. doi: 10.48550/arxiv.2303.10512. URL https://doi.org/10.48550/arxiv.2303.10512.

[15] Changhai Zhou, Shijie Han, Lining Yang, Yuhua Zhou, Xu Cheng, Yibin Wang, and Hongguang Li. Rankadaptor: Hierarchical rank allocation for efficient fine-tuning pruned llms via performance model. 2025. doi: 10.18653/v1/2025.findings-naacl.321. URL https://doi.org/10.18653/v1/2025.findings-naacl.321.