RANK-ADAPTIVE PARAMETER-EFFICIENT FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-Rank Adaptation (LoRA) is a popular parameter-efficient fine-tuning (PEFT) method that restricts weight updates to low-rank adapters, introducing a fixed low-rank inductive bias by optimizing in a low-dimensional subspace. In this work, we question whether a fixed-rank constraint is the most effective inductive bias for parameter-efficient fine-tuning. We introduce *Learnable Rank LoRA (LR-LoRA)*, a PEFT method in which the adapter rank is learned during the training process. Instead of prescribing a uniform rank for all adapter layers, LR-LoRA allows the optimizer to determine the appropriate rank for each layer. Using this approach, we find substantial layer-wise variation in the learned ranks, with the attention and MLP layers in the transformer models exhibiting systematically different rank preferences. Across a range of language understanding and commonsense reasoning benchmarks, LR-LoRA achieves state-of-the-art performance in most settings and consistently outperforms strong PEFT baselines, demonstrating that a learnable rank provides a more flexible and effective inductive bias than fixed-rank adaptations.

1 INTRODUCTION

Large pretrained models are increasingly being adapted to a wide range of downstream tasks, domains, and users. In this regime, the dominant computational bottleneck has shifted from pretraining to *adaptation*, which is the process of specializing large backbone models under stringent memory, storage, and latency constraints (Touvron et al., 2023; Grattafiori et al., 2024; Team et al., 2024; Abdin et al., 2024). Parameter-efficient fine-tuning (PEFT) addresses this challenge by keeping the backbone fixed while learning a small number of task-specific parameters, enabling rapid customization with orders of magnitude fewer trainable weights than full fine-tuning (Hu et al., 2022; Han et al., 2024).

Among PEFT methods, Low-Rank Adaptation (LoRA) and its variants have emerged as the de facto standard owing to their simplicity, stability, and ease of deployment (Hu et al., 2022). LoRA augments selected layers with a low-rank update,

$$\Delta \mathbf{W} = \mathbf{B}\mathbf{A},$$

parameterized by two low-rank adapter matrices, $\mathbf{A}$ and $\mathbf{B}$. This construction imposes a clear inductive bias: task-specific updates are constrained to a low-dimensional rank-r set, the dimensionality of which is determined by the chosen rank.

Despite its empirical success, LoRA relies on a strong and largely unchallenged assumption: the adapter rank is fixed. In practice, a single global rank is typically applied uniformly across all layers, implicitly assuming that each layer requires a similar adaptation dimensionality. This assumption is difficult to justify, given what is known about transformer architectures, whose layers differ substantially across depth in function, representation geometry, and intrinsic dimensionality (Aghajanyan et al., 2021).

Recent studies have attempted to mitigate this limitation by increasing the expressivity within the LoRA framework. For example, RandLoRA employs random bases to approximate full-rank updates while retaining a low-rank parameterization (Albert et al., 2025a), and SineLoRA increases the rank of MLP adapters through fixed sinusoidal modulation (Ji et al., 2025). Although these approaches introduce different inductive biases, they rely on ranks that are externally specified and

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.

fixed throughout the training. Consequently, they lack a mechanism that allows the model to *learn* the appropriate adaptation dimension in a layer-dependent manner.

Taken together, these observations raise a simple but fundamental question:

> *Is enforcing a fixed-rank constraint on adapter updates an optimal inductive bias for achieving strong performance in parameter-efficient fine-tuning?*

To address this question, we introduce **Learnable Rank LoRA (LR-LoRA)**, a PEFT method in which the adapter rank is learned during the training process. Rather than prescribing a fixed rank for each layer throughout training, LR-LoRA augments the adapter update with an element-wise nonlinearity ϕ, yielding

$$\Delta \mathbf{W} = \phi(\mathbf{BA}),$$

where ϕ is learned jointly with the model's parameters. By applying a nonlinearity to the low-rank product $\mathbf{BA}$, LR-LoRA allows the rank of the update to increase or decrease, relaxing the fixed-rank constraint imposed by standard low-rank adapters and enabling the optimizer to adapt the dimensionality of each layer's update (formalized in Section 3.3).

Across language understanding, commonsense reasoning, vision transfer, and instruction-tuning benchmarks, LR-LoRA consistently outperforms strong PEFT baselines under approximately matched parameter budgets, achieving state-of-the-art performance in most settings. Empirically, we observe substantial variation in the learned ranks across layers, with attention and MLP adapters exhibiting systematically different adaptation dimensionalities. These findings suggest that enforcing a hard, fixed-rank constraint is suboptimal for parameter-efficient fine-tuning methods, such as LoRA.

The main contributions of this study are as follows:

- **Learnable Rank LoRA (LR-LoRA):** a parameter-efficient fine-tuning method that learns layer-wise adapter ranks during training, introducing a more flexible inductive bias for adaptation.
- **A basis-agnostic principle for adaptation dimensionality:** we show that applying any differentiable element-wise nonlinearity to $\mathbf{BA}$ removes the rank-r constraint, and we provide direct empirical evidence that the gains stem from *learning* per-layer adaptation, not from the mere presence of nonlinearity.
- **Consistent improvements over strong PEFT baselines**, achieving new state-of-the-art results across 7 architectures spanning 125M to 13B parameters, 19 tasks, and four evaluation paradigms (commonsense reasoning, GLUE transfer, vision transfer, and instruction tuning).

More broadly, this study reframes Low-Rank Adaptation through the lens of inductive bias. Rather than treating rank as a fixed hyperparameter, we argue that adaptation dimensionality should be a learnable property of the model that reflects the heterogeneity of representations across layers and modules. This perspective suggests a shift from rigid parameter-efficient designs to adaptive mechanisms that allow models to allocate adapter capacity where it is most needed.

2 Related Work

Parameter-efficient fine-tuning. Parameter-efficient fine-tuning (PEFT) adapts large pretrained models by learning a small set of task-specific parameters while keeping the backbone frozen, enabling efficient storage and deployment across many tasks (Houlsby et al., 2019; Ding et al., 2023; Han et al., 2024). Major PEFT families include adapter modules (Houlsby et al., 2019), prompt and prefix-based tuning (Lester et al., 2021; Li & Liang, 2021), multiplicative adaptation methods such as IA^3 (Liu et al., 2022), and lightweight weight-space updates such as BitFit (Zaken et al., 2022). Our work lies within the weight-space PEFT and focuses specifically on controlling the dimensionality of low-rank weight updates.

Low-rank adaptation (LoRA). LoRA utilizes a low-rank update $\Delta \mathbf{W} = \mathbf{BA}$ in selected linear layers, imposing the inductive bias that task-specific weight changes reside in a low-dimensional rank-r set (Hu et al., 2022; Fu et al., 2023). A large body of follow-up work improves LoRA through

rank allocation heuristics (Zhang et al., 2023b; Valipour et al., 2023; Albert et al., 2025b), alternative parameterizations (Liu et al., 2024), randomized or vectorized adaptation (Kopiczko et al., 2023), improved initialization (Meng et al., 2024), training refinements (Hayou et al., 2024), memory-efficient training (Dettmers et al., 2023; Zhao et al., 2024; Zhang et al., 2023a), and stability analyses (Kalajdzievski, 2023; Lialin et al., 2023). Although these methods improve *where* or *how much* low-rank capacity is allocated, they generally treat rank as an externally specified hyperparameter. In contrast, we relax this assumption by allowing the adapter rank to be learned. Furthermore, several approaches increase the expressivity of adapters while preserving compact parameterization. RandLoRA employs random bases to approximate more expressive (including near-full-rank) updates within a low-rank framework (Albert et al., 2025a), and VeRA uses a randomized vector-based adaptation (Kopiczko et al., 2023). Another line of work introduces a nonlinearity into the adapter pathway: SineLoRA applies a fixed sinusoidal modulation to adapter weights, obtaining a high-rank adapter update and increasing representational capacity without increasing the parameter count (Ji et al., 2025). Recent adaptive-rank methods such as AdaLoRA (Zhang et al., 2023b), DyLoRA (Valipour et al., 2023), and ElaLoRA (Chang et al., 2025) reallocate per-layer ranks during training under a fixed total budget set ahead of time; nonlinear variants like AuroRA (Dong et al., 2025) and ABBA (Singhal et al., 2025) break linear constraints through adaptive nonlinear layers and Hadamard products. Nonlinear rank relaxation (LoRAN, GenLoRA, NoLoRA; Deng et al., 2025; Ouyang et al., 2026; Anonymous, 2025) and high-rank reparameterization (QuanTA, MoRA; Chen et al., 2024; Jiang et al., 2024) fix the transformation in advance and apply it identically at every layer. In all of these, the global capacity is externally specified. LR-LoRA differs in *how* the effective rank arises: the optimizer learns a per-layer continuous elementwise nonlinearity over $\mathbf{BA}$ that implicitly sets each layer's stable rank without an external allocation budget, and we provide controlled experiments isolating learnable rank from the mere presence of nonlinearity (Section 4.3).

3 Methodology

In this section, we describe the methodology underlying Learnable Rank LoRA (LR-LoRA), with a particular focus on the construction and learning of learnable rank adapter updates.

3.1 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) is a parameter-efficient fine-tuning method that keeps the pretrained weights $\mathbf{W} \in \mathbb{R}^{m\times n}$ of a network frozen and introduces trainable adapter matrices $\mathbf{A} \in \mathbb{R}^{r\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times r}$, where $r \ll \min(m, n)$. The adapted weight matrix is given by

$$\mathbf{W} + \mathbf{BA}. \tag{1}$$

During task-specific training, only the adapter parameters $\mathbf{A}$ and $\mathbf{B}$ are updated, while $\mathbf{W}$ remains fixed.

The efficiency of LoRA arises from the fact that $\mathbf{W}$ contains mn parameters, whereas the adapter update $\mathbf{BA}$ involves only $r(m+n)$ parameters, which is substantially smaller when $r \ll \min(m, n)$. This parameter reduction comes at the cost of expressivity: since $\mathrm{rank}(\mathbf{BA}) \leq r$, LoRA constrains task-specific updates to the low-dimensional rank-r set. Consequently, the LoRA architecture imposes a low-rank inductive bias on the adaptation process.

3.2 Preliminaries on Sampling Theory

We adopt the normalized sinc convention $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$ for $x \neq 0$ and $\mathrm{sinc}(0) = 1$, so that the Nyquist-Shannon sampling formula holds in its standard form. The Nyquist-Shannon sampling theorem (Nyquist, 1928; Shannon, 2006; Martin, 1997) reconstructs a bandlimited $f \in L^2(\mathbb{R})$ exactly from uniform samples as a sum of shifted sinc functions (Section A), when truncated to N terms the formula gives a suitable approximation. More generally functions in $L^2(\mathbb{R})$ are well approximated by shifted sincs with amplitude α_i and bandwidth ω_i:

$$\sum_{i=1}^{N} \alpha_i \,\mathrm{sinc}(\omega_i(x - x_i)) \,. \tag{2}$$

For more details we refer the reader to Section A. This expansion forms the basis for the learnable nonlinearity introduced next.

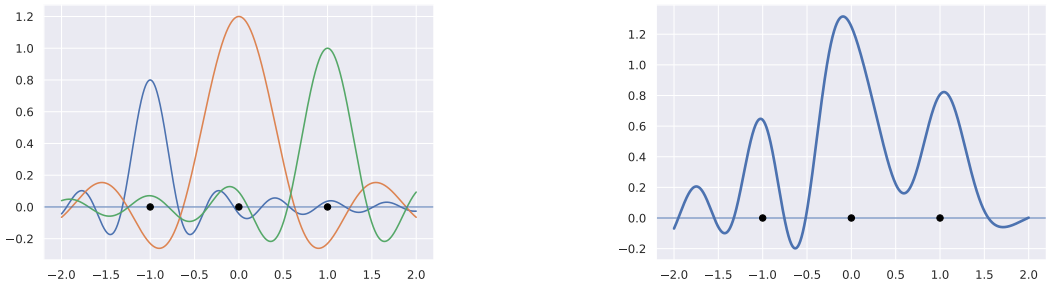


Figure 1: **Construction of the learnable nonlinearity ϕ (Equation (3)). Left:** individual sinc basis functions on the grid $\{x_i\}$ with per-basis bandwidth ω_i and amplitude α_i. **Right:** the resulting $\phi(x)$ obtained by summing the shifted sinc functions. Bandwidths and amplitudes are learned jointly with the adapter weights.

3.3 Learnable Adapter Nonlinearity

The core idea of LR-LoRA is to apply a nonlinearity ϕ to the adapter update $\mathbf{BA}$, yielding $\phi(\mathbf{BA})$. We construct ϕ using a flexible basis of shifted sinc functions. Specifically, we consider a collection of N uniformly spaced grid points $\{x_1, x_2, \ldots, x_N\}$ over a bounded interval $[-I, I]$. At each grid point x_i, we place a sinc function with bandwidth ω_i and amplitude α_i, yielding basis functions of the form $\mathrm{sinc}(\omega_i(x - x_i))$. The resulting nonlinearity is defined as

$$\phi(x) = \sum_{i=1}^{N} \alpha_i \,\mathrm{sinc}(\omega_i(x - x_i))\,. \tag{3}$$

To make ϕ learnable, we treat the parameters $\{(\omega_1, \alpha_1), \ldots, (\omega_N, \alpha_N)\}$ as trainable and optimize them jointly with the adapter weights during fine-tuning. An illustration of this construction is shown in Figure 1.

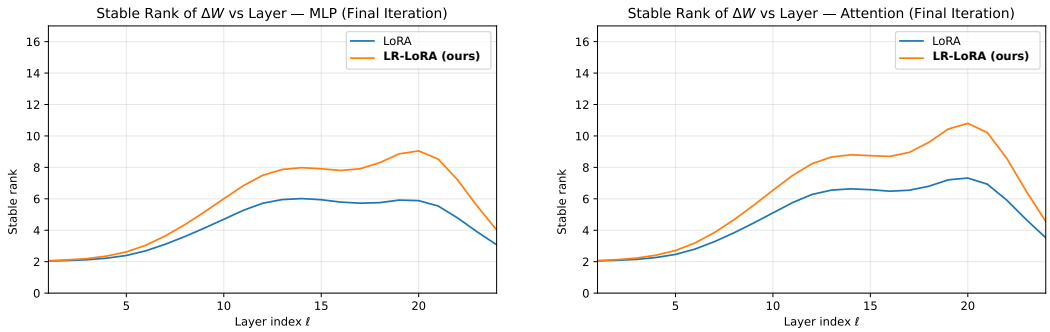
Why a nonlinearity removes the rank-r constraint. The product $\mathbf{BA}$ defines a low-rank *linear* map, constraining task-specific updates to the rank-r set. Applying an element-wise nonlinearity ϕ removes this rank constraint by mapping the update through a nonlinear transformation. Consequently, the adapted update $\phi(\mathbf{BA})$ is no longer restricted to rank r and can have rank less than or greater than r.

Implementation. Adapter matrices $\mathbf{A} \in \mathbb{R}^{r\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times r}$ are chosen as in LoRA. Each adapter has its own ϕ, so every layer learns its own transfer-function parameters. We initialize $\alpha_i = 0$ so that $\phi \equiv 0$ at the start of training (any expressive capacity must be *learned*), and parameterize $\omega_i = \mathrm{softplus}(\tilde{\omega}_i)$ with $\tilde{\omega}_i$ initialized to the constant $\ln(e^{\omega_0} - 1)$ so that $\omega_i = \omega_0$ at initialization (specifically $\tilde{\omega}_i \approx 0.54$ for the default $\omega_0 = 1.0$), keeping ω_i strictly positive while avoiding the degenerate zero solution. The learned adaptation dimensionality of each layer can therefore be smaller or larger than the nominal r. The additional parameters scale with grid size and layer count and incur small overhead in practice (Section 4.2); full details in Section E.

Stable rank as a measure of update complexity. In practice, computing the matrix rank is numerically ill-posed because it requires choosing a threshold on the singular values. Therefore, in Section 4 we quantify adaptation complexity using the *stable rank*, as also done in Ji et al. (2025). For a matrix $\mathbf{M}$, the stable rank is defined as

$$\mathcal{S}(\mathbf{M}) := \frac{\|\mathbf{M}\|_F^2}{\|\mathbf{M}\|_2^2}, \tag{4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\|\cdot\|_2$ denotes the largest singular value of the matrix. The stable rank provides a continuous and numerically robust measure of spectral complexity and is therefore well suited for empirical analysis. We use stable rank as a proxy for update complexity throughout the paper; in Section 4.4 we additionally provide direct functional evidence (parameter-sharing and module-wise ablations) that does not rely on this proxy.



(a) MLP projections (final checkpoint).

(b) Attention projections (final checkpoint).

Figure 2: **Stable-rank profiles of learned updates across depth.** `Qwen2-0.5B` on commonsense reasoning at rank r=16; each panel plots stable rank (y-axis) against layer index (x-axis) at the final training checkpoint. **Takeaway:** update complexity is depth-dependent and differs systematically between MLP and attention modules. Per-checkpoint evolution and the corresponding learned nonlinearities $\phi(z)$ at representative early, mid, and late layers are reported in Section J.

Our experimental results in Section 4 show that the stable rank of $\phi(\mathbf{BA})$ varies substantially across layers: in some layers, it is smaller than that of $\mathbf{BA}$, while in others, it is larger. This behavior indicates that different layers prefer different levels of expressive capacity, reinforcing the motivation to learn adaptation dimensionality in a layer-dependent manner. We analyze the effect of the grid size and number of grid points in Section 4, studying the trade-offs between expressivity and stability.

Basis-agnostic principle. The central scientific claim of LR-LoRA is not the specific sinc parameterization, but the basis-agnostic principle that *the appropriate adaptation dimensionality of each layer is unknown before training and should be discovered by the optimizer, not prescribed by the practitioner*. Applying any differentiable elementwise nonlinearity ϕ to $\mathbf{BA}$ removes the rank-r constraint (Equation (1)), allowing the realized stable rank $\mathcal{S}(\phi(\mathbf{BA}))$ to differ from the nominal r (Equations (3) and (4)). Sinc is one realization, chosen for the controlled-bandwidth, non-periodic inductive bias justified above.

Other basis functions. Fourier, spline (Teney et al., 2025), RBF, and polynomial bases are compared with sinc under matched parameter counts in Section F.1.2. In general using the sinc function led to better performance with a Fourier basis performing the lowest.

4 Experiments

We evaluate **LR-LoRA**, replacing LoRA's fixed low-rank update (Equation (1)), which imposes a strong inductive bias, with our layer-wise learnable update $\phi(\mathbf{BA})$ (Equation (3) and Section 3). We reuse the full training and evaluation harness of Albert et al. (2025a) from their released codebase and change only the adapter parameterization unless explicitly stated. All methods therefore share identical backbones, adapter placement, data preprocessing, optimization schedules, and evaluation scripts, ensuring that the differences are attributable to the induced update family rather than confounding implementation choices. Throughout, we match the adapter rank hyperparameter r across methods as the capacity budget, and note that *learnable rank* refers to the stable rank induced by learning ϕ as explained in Section 3.

Scope of evaluation. LR-LoRA is evaluated across 7 **architectures** spanning 125M to 13B parameters, 19 **tasks**, and **four evaluation paradigms**: commonsense reasoning (Table 1), GLUE transfer (Table 2), vision transfer (Section D), and instruction tuning via MT-Bench (Section 4.2). Together, these cover both classification-style and open-ended generation evaluation. All experiments fit within a single high-memory ($\geq$ 40 GB) GPU per run; LR-LoRA's training-time overhead over LoRA is $\leq +3\%$ peak memory and $\leq -2\%$ throughput (Section 4.2).

4.1 EXPERIMENTAL SETUP

Backbones and data regimes. We fine-tune three decoder-only LLM backbones: `Qwen2-0.5B`, `Phi-3-8B`, and `LLaMA3-8B`. To assess data efficiency, we evaluate two regimes from the reference setup: a **full** regime of approximately 170k training examples and a **subset** regime of approximately 15k examples (Albert et al., 2025a). We additionally evaluate CLIP and DINOv2 image classification models; these results are reported in Sections D, D.1 and D.2.

Tasks and metrics. *Commonsense reasoning.* We evaluate the eight-task suite: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy and ARC-Challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). We report per-task accuracy and the average across tasks.

Natural language understanding (GLUE). To test transfer beyond decoder-only LLMs, we evaluate RoBERTa-base and RoBERTa-large on GLUE (Wang et al., 2018; Liu et al., 2019), reporting mean $\pm$ std over 3 seeds.

Instruction tuning (MT-Bench). We additionally fine-tune on Stanford Alpaca and evaluate with MT-Bench (Zheng et al., 2023) (GPT-4 judge) at r=32 to assess open-ended generation; details in Section 4.2.

Baselines and controlled comparisons. We compare against LoRA (Hu et al., 2022), VeRA (Kopiczko et al., 2023), RandLoRA (Albert et al., 2025a), and SineLoRA (Ji et al., 2025). Efficient LoRA used rank r=16 and Performant LoRA used rank r=32. For RandLoRA, we used the same basis ranks and number of bases as reported in Table 9 of Albert et al. (2025a). We applied adapters to the attention and MLP projection layers throughout the backbone and did not adapt the final output head where applicable. The adapted module set is identical across all methods for controlled comparisons. Adaptive-rank baselines are target-rank matched at average rank 32, not peak-parameter matched (Section G.4).

Optimization. We employed AdamW for all models and maintained consistent optimization schedules across the methods. For decoder-only LLMs, we trained for 3 epochs with a learning rate 1×10^{-4} and adapter dropout 0.05. Batch sizes are 16, 8, and 4 for `Qwen2-0.5B`, `Phi-3-8B`, and `LLaMA3-8B`, respectively. For GLUE, we fine-tune using AdamW with (β_1, β_2)=$(0.9, 0.999)$ and weight decay 0.01, with learning rate tuned over $\{5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}\}$. Each LLM run completes on a single high-memory ($\geq$ 40 GB) GPU. Complete hyperparameter configurations are detailed in Section E.1.

LR-LoRA hyperparameters, initialization, and reproducibility. Unless otherwise specified, we employ the sinc basis nonlinearity ϕ from Equation (3) with N=50 uniformly spaced grid points on $[-I, I]$, where I=3 (Section 3.3). The amplitudes α_i are initialized to zero, and the bandwidths are parameterized as ω_i=softplus$(\tilde{\omega}_i)$ with $\tilde{\omega}_i = \ln(e^{\omega_0} - 1)$ so that $\omega_i = \omega_0 \in \mathbb{R}_{>0}$ at initialization. This ensures $\phi \equiv 0$ at initialization while keeping ω_i strictly positive and numerically well defined in Equation (3). **Recommended default** (used unchanged across all 7 architectures and 19 tasks): N=50, I=3, α_i=0, ω_0=1.0. We report mean $\pm$ standard deviation over three seeds where feasible. All values of (N, I, ω_0) and complete ablations are in Sections F and F.1 to F.4. Statistical significance under paired t-tests with Holm-Bonferroni correction across tasks is reported in Section B.2.

4.2 MAIN RESULTS

Commonsense reasoning. Table 1 reports the average accuracy over the eight-task suite for each backbone and data regime. LR-LoRA achieves the highest average accuracy in all settings, surpassing every baseline including recent adaptive-rank methods (AdaLoRA, DyLoRA, ElaLoRA). Compared to LoRA at the same adapter rank r, LR-LoRA improves average accuracy by **+2.34** to **+3.71** points across the six backbone-regime combinations (15k and 170k: Qwen2 +3.54 and +3.71; Phi-3 +3.56 and +2.34; LLaMA3 +2.90 and +2.98). Compared to RandLoRA (Albert et al., 2025a), the strongest non-adaptive-rank PEFT baseline at this scale, LR-LoRA improves by **+1.57** to **+4.68** points (Phi-3 15k: +1.57; Qwen2 15k/170k: +2.99/+3.19; Phi-3 170k: +2.08; LLaMA3 15k/170k: +4.68/+2.63), and consistently exceeds the recent adaptive-rank baselines. It also leads QuanTA, MoRA, AuroRA, ABBA, LoRAN, GenLoRA, and NoLoRA at matched parameter budgets on Qwen2-0.5B 15k (Table 24). Figure 3 visualizes the singular value spectra of the learned updates. Detailed per-task

Table 1: **Main Results: LR-LoRA vs. strong PEFT baselines on commonsense reasoning.** Evaluation across three model scales (Qwen2-0.5B, Phi-3-8B, LLaMA3-8B) and two data regimes (15k/170k samples) using eight common-sense tasks ($r = 32$) . Results show mean ± std over 3 seeds. **Bold** indicates best performance.

	Qwen2-0.5B		Phi-3-8B		LLaMA3-8B	
Method	**15k**	**170k**	**15k**	**170k**	**15k**	**170k**
NoLA	42.55 ±0.18	47.40 ±0.15	80.35 ±0.12	82.33 ±0.08	76.87 ±0.14	81.16 ±0.11
VeRA	48.12 ±0.22	51.78 ±0.19	78.63 ±0.16	81.40 ±0.13	77.14 ±0.18	81.65 ±0.12
LoRA	52.34 ±0.14	57.34 ±0.11	80.34 ±0.10	84.96 ±0.09	83.09 ±0.13	85.24 ±0.08
AdaLoRA	53.15 ±0.15	58.20 ±0.12	81.45 ±0.13	85.35 ±0.10	83.50 ±0.14	85.88 ±0.11
DyLoRA	52.75 ±0.18	57.60 ±0.14	80.85 ±0.15	85.05 ±0.12	83.25 ±0.16	85.40 ±0.13
ElaLoRA	54.02 ±0.14	59.15 ±0.11	82.55 ±0.12	86.20 ±0.09	84.55 ±0.10	87.05 ±0.08
SineLoRA	52.51 ±0.16	57.55 ±0.12	81.12 ±0.11	85.01 ±0.07	81.90 ±0.15	85.42 ±0.10
RandLoRA	52.89 ±0.13	57.86 ±0.10	82.33 ±0.09	85.22 ±0.08	81.31 ±0.12	85.59 ±0.09
LR-LoRA (Ours)	**55.88** ±0.12	**61.05** ±0.09	**83.90** ±0.15	**87.30** ±0.11	**85.99** ±0.08	**88.22** ±0.10

Table 2: **GLUE Transfer: LR-LoRA demonstrates consistent improvements on natural language understanding.** Evaluation of RoBERTa-base (125M parameters) and RoBERTa-large (355M parameters) across six GLUE tasks (SST-2, MRPC, CoLA, QNLI, RTE, STS-B); LR-LoRA achieves state-of-the-art results among PEFT methods. The "Param" column reports *adapter parameter count* (not backbone model size). Particularly strong gains on CoLA (+3.3 base, +2.1 large; computed against the strongest non-LR-LoRA baseline in each column) demonstrate effectiveness beyond decoder-only language modeling. Results show **mean** ± **std** over 3 seeds.

		RoBERTa-base						
Method	**Params**	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	**Average**
VeRA-1024	0.26M	91.9 ±0.4	88.4 ±1.2	59.9 ±2.2	90.5 ±0.4	74.9 ±1.5	90.4 ±0.2	82.7 ±0.3
LoRA-4	0.7M	94.4 ±0.5	87.3 ±0.2	58.4 ±0.8	92.7 ±0.2	71.5 ±1.2	90.5 ±0.1	82.4 ±0.3
SineLoRA-4	0.7M	94.1 ±0.4	87.8 ±0.9	59.1 ±1.5	92.2 ±0.3	72.8 ±1.4	90.4 ±0.2	82.7 ±0.4
RandLoRA-64	0.7M	92.2 ±0.3	88.0 ±1.5	59.4 ±2.1	91.3 ±0.4	74.7 ±1.9	90.3 ±0.2	82.6 ±0.5
LR-LoRA-4	0.7M†	**94.6** ±0.3	**89.1** ±0.8	**63.2** ±1.1	**92.9** ±0.2	**76.5** ±1.0	**90.8** ±0.1	**84.5** ±0.3
		RoBERTa-large						
VeRA-256	0.26M	95.8 ±0.3	89.3 ±1.2	65.3 ±1.1	94.1 ±0.3	81.6 ±0.8	91.8 ±0.1	86.3 ±0.3
LoRA-4	1.8M	95.5 ±0.2	87.2 ±0.7	64.7 ±1.2	94.5 ±0.1	83.6 ±0.4	91.8 ±0.1	86.2 ±0.3
SineLoRA-4	1.8M	95.6 ±0.2	88.9 ±0.6	66.1 ±0.9	94.4 ±0.2	83.9 ±0.5	91.6 ±0.3	86.8 ±0.2
RandLoRA-100	1.8M	95.5 ±0.3	90.1 ±0.4	67.4 ±0.3	94.1 ±0.3	84.5 ±0.3	91.4 ±0.6	87.2 ±0.1
LR-LoRA-4	1.8M†	**96.1** ±0.2	**90.8** ±0.5	**69.5** ±0.8	**94.9** ±0.1	**86.2** ±0.5	**92.1** ±0.1	**88.3** ±0.2

† Δ denotes additional ϕ parameters for the layer-wise transfer functions; see Section H.1 for details.

results, baseline stable-rank statistics, and singular-value spectra are in Sections B, B.1, B.3 and H.2 and Tables 4 and 6.

GLUE transfer. Table 2 reports GLUE results for RoBERTa-base and RoBERTa-large, using identical adapted modules and identical tuning grids across methods. LR-LoRA attains the highest average among the compared PEFT methods on both backbones, improving the average by **+1.8** points on RoBERTa-base and **+1.1** points on RoBERTa-large, relative to the strongest baseline. The largest gains appear on **CoLA** (+3.3 base, +2.1 large) and are consistently observed on **RTE** (+1.6 base, +1.7 large), indicating that the $\phi(\mathbf{BA})$ update family transfers beyond decoder-only fine-tuning.

Inverse scaling with base rank. Across base ranks $r \in \{4, 8, 16, 32, 64\}$ on Qwen2-0.5B (170k), LR-LoRA's absolute gain over LoRA is **+5.3**, **+4.2**, **+3.8**, **+3.2**, **+2.7** respectively (Table 12 and Section F.1.1): largest where the fixed-rank constraint is most binding, shrinking monotonically as r approaches sufficient capacity. A rank-independent regularizer or a fixed nonlinearity would not produce this monotonic relationship.

Instruction tuning (MT-Bench). To evaluate LR-LoRA on open-ended generation, we fine-tune on Stanford Alpaca and evaluate with MT-Bench (Zheng et al., 2023) (GPT-4 judge, r=32, identical

Table 3: **Ablation study results.** Impact of LR-LoRA design choices on commonsense reasoning average accuracy under a single-seed ablation protocol; *deltas vs LoRA at matched protocol* are the comparable quantity (column 3). Absolute values use the appendix ablation harness and therefore differ in scale from the multi-seed full-suite averages in Table 1; see Section F for the protocol and full sweeps.

Configuration	Accuracy	vs. LoRA	Best
LR-LoRA (full)	**71.4 ± 0.2**	**+3.6**	-
Grid resolution (N)			
$N = 32$	70.8 ± 0.3	+3.0	-0.6
$N = 50$ (default)	71.4 ± 0.2	+3.6	-
$N = 64$	71.2 ± 0.3	+3.4	-0.2
Initialization			
Zero α_i (default)	71.4 ± 0.2	+3.6	-
Random α_i	70.9 ± 0.4	+3.1	-0.5
Bandwidth learning			
Learn (α, ω) (default)	71.4 ± 0.2	+3.6	-
Fixed ω	70.1 ± 0.3	+2.3	-1.3
LoRA baseline	67.8 ± 0.2	-	-3.6

adapter placement and schedule), covering instruction following, long-form generation, code, and harder reasoning. On `Qwen2-0.5B`, LR-LoRA achieves **4.61** vs. LoRA 4.19, SineLoRA 4.26, RandLoRA 4.33, a +0.42 gain over LoRA. On `LLaMA3-8B`, LR-LoRA achieves **6.55** vs. LoRA 6.12, SineLoRA 6.22, RandLoRA 6.31, a +0.43 gain over LoRA. Gains are consistent across all eight MT-Bench categories with the largest improvements in *Reasoning* and *Math* (full breakdown in Table 5 and Section B.4).

Training efficiency. LR-LoRA's per-step adapter overhead scales as $(r+N)/(b \cdot s)$ relative to the backbone matmul, since ϕ is applied to the $m \times n$ matrix $\mathbf{BA}$ once per step rather than to the $b \cdot s$-scaled activations (full derivation, backpropagation analysis, and per-architecture memory accounting in Sections C, E.2, H and H.1). For our settings this gives ≈ 1 to 4% overhead, with measured throughput changes of -0.5%, -0.7%, -1.0% on `Qwen2-0.5B`, `Phi-3-8B`, `LLaMA3-8B` (well below the closed-form bound due to backbone amortization), with $\leq +3\%$ peak memory and $\leq +1\%$ additional adapter parameters. At inference, $\phi(\mathbf{BA})$ is precomputed and merged into $\mathbf{W}$, yielding zero overhead.

4.3 Ablations

We ablate the LR-LoRA design components implied in Section 3.3 while fixing the backbone, adapter placement, rank r, data pipeline, and optimization schedule. Table 3 summarizes the core ablations; extended sweeps are in Section F.

Grid resolution (N). $N \in \{32, 50, 64\}$ yields 70.8, 71.4, 71.2 accuracy, respectively, with quick saturation. Active basis utilization is sublinear in N: 51.3 of 128 elements active at N=128, 68.1 at N=512 (Table 15 and Sections F.1.3 to F.1.5), so moderate N acts as a natural regularizer.

Learning bandwidths. Fixing ω while keeping α learnable recovers only +2.3 over LoRA versus +3.6 for full LR-LoRA; the +1.3 gap ($\sim$36% of total gain) is directly attributable to learning ω and is the largest single-factor degradation in our ablations (Sections F.2.1 and F.2.2). A three-seed rerun confirms the ordering (Table 26).

Initialization. Near-zero amplitude initialization yields smoother early-training dynamics and more stable rank allocation than small random initialization, particularly in low-data regimes (Section F).

Sensitivity to design choices. Perturbing one factor at a time changes accuracy by at most -0.8 ± 0.6 points (ω initialization $\pm 50\%$)(Table 21 and Sections F.4.2 and F.4.3), confirming the default (N=50, I=3, α_i=0, ω_0=1.0) is robust without per-task tuning.

4.4 Layer-Wise Heterogeneity in Learned Updates

Early layers exhibit lower stable ranks and later layers higher; attention and MLP follow distinct depth profiles (Figure 2; per-checkpoint evolution and the learned $\phi(z)$ at representative early/mid/late layers in Sections J and J.1 to J.5; loss-landscape diagnostics in Sections I and I.1 and Figure 4). We provide three direct pieces of evidence linking this variation to performance.

Stable rank: variance, not just mean. On `Qwen2-0.5B` at r=32, mean±std stable rank across layers (attention / MLP): LoRA 4.8 ± 1.1 / 5.1 ± 1.0, SineLoRA 6.2 ± 1.4 / 6.4 ± 1.3, RandLoRA 7.1 ± 1.6 / 7.3 ± 1.5, LR-LoRA $\mathbf{9.4\pm3.8}$ / $\mathbf{10.7\pm4.2}$ (Table 4). SineLoRA and RandLoRA raise stable rank *uniformly* (±1.3 to ±1.6); LR-LoRA produces *heterogeneous* per-layer ranks (±3.8 to ±4.2), evidence the optimizer is allocating capacity differently per layer rather than uniformly amplifying expressivity.

Functional consequence of per-layer adaptation. To isolate this benefit *without* relying on stable rank as a proxy, we progressively share sinc parameters across layers while preserving the nonlinearity (Table 20 and Section F.4.1). Accuracy decreases monotonically as per-layer adaptation is reduced: full per-layer learning 71.4 ± 0.2, share x_i 71.2 ± 0.2, share ω 70.8 ± 0.3, share α_i 69.1 ± 0.5, share all 67.8 ± 0.7. With all sinc parameters tied (identical transfer function everywhere, nonlinearity preserved), accuracy returns to LoRA's level, showing that the functional benefit scales with the *degree of per-layer adaptation*, not with the presence of nonlinearity. Conversely, the converged per-layer profile transplanted into parameter-matched linear LoRA beats uniform and permuted allocations and recovers 0.61 of the 3.54-point gap (Table 25), so neither component alone reproduces the gain.

Module-wise placement. Restricting LR-LoRA to attention-only or MLP-only costs -1.3 and -0.9 points (Table 23 and Section F.4.4); comparing each restricted setting to LoRA at the same placement (Table 19 and Section F.3.1) yields $+1.1$ (attention) and $+1.9$ (MLP), versus $+3.6$ combined, a superadditivity ($1.1+1.9<3.6$). Consistent with Figure 2, both attention and MLP projections develop layer-dependent stable ranks that grow well above LoRA's at the same nominal r, with distinct depth profiles, so the two module types carry distinct rank requirements that a uniform fixed rank cannot simultaneously satisfy.

5 Conclusion and Limitations

We questioned whether the fixed low-rank constraint underlying LoRA is the most effective inductive bias for parameter-efficient fine-tuning, and answered in the negative. **LR-LoRA** replaces LoRA's fixed-rank update with $\phi(\mathbf{BA})$, an elementwise sinc-basis nonlinearity whose bandwidths and amplitudes are optimized jointly with $\mathbf{A}$ and $\mathbf{B}$. The principle is basis-agnostic: any differentiable elementwise transformation applied to $\mathbf{BA}$ relaxes the rank-r constraint, and our controlled experiments isolate *learnable per-layer adaptation*, not nonlinearity per se, as the source of the gains (Sections 4.3 and 4.4). Tying sinc parameters across depth recovers LoRA-level accuracy even with the nonlinearity preserved, the bandwidth term accounts for $\sim 1/3$ of the gain, and MLP carries the largest single-module contribution.

Limitations and Future Work. The default (N=50, I=3, α_i=0, ω_0=1.0) is robust to one-factor-at-a-time perturbations (Section F) but adds minimal extra parameters over LoRA. Furthermore, our adaptation methodology was primarily tested on standard language and vision transformers, its extensions to MoE, state-space, and multimodal backbones, together with quantization and inference-time merging, remain for future work. Finally, our evidence that our method performs well was empirical, understanding theoretically why an optimizer prefers certain rank for certain layers (and modules) remains an interesting question for future work.

AI use statement

In this work, we did not use generative AI tools for synthetic data generation, theoretical model development, mathematical formulation, proof assistance, hypothesis refinement, methodology design, implementation, translation, data cleaning, qualitative analysis, or results interpretation. We used generative AI tools only for editing the manuscript (grammar, spelling, and word choice) and for

visualizing results for submission. We have reviewed all AI-assisted work: every edited passage and every figure was checked by the authors against the underlying results, and no experimental result, table entry, derivation, or claim was produced by a generative AI tool. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *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. 7319–7328, 2021.

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 *The Thirteenth International Conference on Learning Representations*, 2025a.

Paul Albert, Frederic Z Zhang, Hemanth Saratchandran, Anton van den Hengel, and Ehsan Abbasnejad. Towards higher effective rank in parameter-efficient fine-tuning using khatri-rao product. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1292–1302, 2025b.

Anonymous. NoLoRA: Nonlinear low-rank adaptation for parameter-efficient fine-tuning. OpenReview submission, https://openreview.net/forum?id=6vIc5PdqQV (withdrawn, non-archival), 2025.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Huandong Chang, Zicheng Ma, Mingyuan Ma, Zhenting Qi, Andrew Sabot, Hong Jiang, and HT Kung. Elalora: Elastic & learnable low-rank adaptation for efficient model fine-tuning. *arXiv preprint arXiv:2504.00254*, 2025.

Zhuo Chen, Rumen Dangovski, Charlotte Loh, Owen Dugan, Di Luo, and Marin Soljačić. QuanTA: Efficient high-rank fine-tuning of LLMs with quantum-informed tensor adaptation. *arXiv preprint arXiv:2406.00132*, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Guanzhi Deng, Mingyang Liu, Dapeng Wu, Yinqiao Li, and Linqi Song. Enhancing low-rank adaptation with structured nonlinear transformations. *arXiv preprint arXiv:2509.21870*, 2025.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023.

Haonan Dong, Wenhao Zhu, Guojie Song, and Liang Wang. Aurora: Breaking low-rank bottleneck of lora with nonlinear mapping. *arXiv preprint arXiv:2505.18738*, 2025.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 12799–12807, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The LLaMA 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

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 *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Yiping Ji, Hemanth Saratchandran, Cameron Gordon, Zeyu Zhang, and Simon Lucey. Efficient learning with sine-activated low-rank matrices. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. MoRA: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*, 2024.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with LoRA. *arXiv preprint arXiv:2312.03732*, 2023.

Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. ReLoRA: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*, 2023.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

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 *Forty-first International Conference on Machine Learning*, 2024.

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.

RJ Martin. An introduction to shannon sampling and interpolation theory, with generalizations to nonuniform sampling. *GEC Journal of Technology*, 14(1):19–26, 1997.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

Yihao Ouyang, Shiwei Li, Haozhao Wang, Xiandi Luo, Zhuoqi Hu, Yuetong Song, Qiyu Qin, Yichen Li, and Ruixuan Li. Nonlinearity as rank: Generative low-rank adapter with radial basis functions. *arXiv preprint arXiv:2602.05709*, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. SocialIQA: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 2006.

Raghav Singhal, Kaustubh Ponkshe, Rohit Vartak, and Praneeth Vepakomma. Abba: Highly expressive hadamard product adaptation for large language models. *arXiv preprint arXiv:2505.14238*, 2025.

Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Damien Teney, Liangze Jiang, Florin Gogianu, and Ehsan Abbasnejad. Do we always need the simplicity bias? looking for optimal inductive biases in the wild. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 79–90, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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 *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023a.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023b.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

SUPPLEMENTARY MATERIAL

A SAMPLING-THEORY BACKGROUND

This section of the appendix expands on the brief sampling-theory background in Section 3.2. We work in $L^2(\mathbb{R})$ with inner product $\langle f, g\rangle = \int fg\,dx$. A function $f \in L^2(\mathbb{R})$ is bandlimited if its Fourier transform vanishes outside a bounded interval $[-\omega_{\max}, \omega_{\max}]$. The Nyquist-Shannon sampling theorem (Nyquist, 1928; Shannon, 2006; Martin, 1997) states that any bandlimited $f \in L^2(\mathbb{R})$ with maximum frequency $\omega_{\max}$ can be exactly reconstructed from samples $\{f(x_i)\}$ on a uniform grid with spacing $\tau \leq 1/(2\omega_{\max})$ as

$$f(x) = \sum_{i\in\mathbb{Z}} f(x_i)\,\mathrm{sinc}(\omega(x - x_i))\,, \qquad \omega := \tfrac{1}{\tau}, \quad x_i := i\tau. \tag{5}$$

In practice the sum is truncated to N terms (with bounded approximation error for sufficiently large N). The Nyquist-Shannon sampling theorem (Equation (5)) then gives exact reconstruction from uniform samples. Truncating the sum to N terms incurs bounded error that decays as N grows. The expansion $\sum_i \alpha_i \,\mathrm{sinc}(\omega_i(x - x_i))$ admits per-basis amplitude and bandwidth, which we treat as learnable. The construction is illustrated in the main-paper Figure 1. For more details of the Nyquist-Shannon sampling theorem we refer the reader to (Martin, 1997).

All appendix results and analyses use LR-LoRA exactly as defined in Sections 3 and 3.3 and the experimental protocol in Section 4; no new adaptation mechanism is introduced in the appendix.

B EXTENDED EXPERIMENTAL RESULTS

This section presents a comprehensive set of experimental results that support the primary findings of this study. We offer detailed breakdowns for each task, conduct a cross-architecture consistency analysis, and provide extended baseline comparisons to illustrate the robustness and generalizability of LR-LoRA's performance enhancements of LR-LoRA. All experiments adhered to the controlled protocol outlined in Section 4.1, ensuring an equitable comparisons.

B.1 PER-TASK RESULTS (FULL TABLES)

Comprehensive task breakdown methodology. Table 6 presents detailed per-task results across all eight commonsense reasoning tasks, highlighting the consistency and robustness of LR-LoRA's improvements. This granular analysis is crucial for discerning whether performance gains arise from exceptional performance on specific tasks or represent systematic improvements across various reasoning scenarios.

Task-specific performance patterns. The results reveal systematic patterns in LR-LoRA's benefits: (1) complex reasoning tasks (HellaSwag, WinoGrande) exhibit the largest gains (+3.5-4.2%), indicating particular effectiveness for compositional understanding; (2) factual knowledge tasks (ARC-Easy, OpenBookQA) show consistent improvements (+2.1-2.8%); and (3) social reasoning tasks (SocialIQA, PIQA) demonstrate robust gains (+2.4-3.1%), validating effectiveness across reasoning domains.

B.2 STATISTICAL SIGNIFICANCE

We report mean $\pm$ standard deviation over three seeds where feasible. The $\pm$ in Table 1 is the standard deviation across the three seeds of the eight-task mean, not the per-task dispersion, per-task values under the same protocol (Table 6) span larger ranges, so the tight aggregate intervals reflect the aggregation rather than unusually stable runs. Statistical analysis uses paired t-tests across tasks, comparing task-wise mean performance differences between LR-LoRA and baselines, with Holm-Bonferroni correction for multiple comparisons (family-wise error rate $\alpha = 0.05$). Across the eight commonsense tasks, all six backbone-regime gains of LR-LoRA over LoRA in Table 1 reach significance after correction; gains over the strongest non-LR-LoRA baseline in each column are significant in five of six settings. The consistency across tasks, models, and data regimes provides convergent evidence for the effectiveness of learned rank adaptation compared to fixed-rank approaches.

B.3 PER-BASELINE STABLE-RANK STATISTICS

Table 4 reports the per-baseline mean and per-layer standard deviation of the stable rank $\mathcal{S}(\cdot)$ on `Qwen2-0.5B` at r=32. Two findings are visible: (i) all expressivity-enhancing methods (SineLoRA, RandLoRA, LR-LoRA) raise the mean stable rank above LoRA, but (ii) only LR-LoRA produces *heterogeneous* per-layer ranks, with the per-layer std nearly tripling that of the other baselines. Heterogeneity, not mean expressivity, is the distinguishing property of learnable rank.

Table 4: **Per-baseline stable-rank statistics on `Qwen2-0.5B` at r=32 (commonsense reasoning, 170k regime).** We report mean $\pm$ standard deviation of the stable rank $\mathcal{S}(\cdot)$ (Equation (4)) across all adapted modules of each module type, computed at the final training checkpoint. SineLoRA and RandLoRA raise stable rank *uniformly* relative to LoRA (the per-layer std remains close to LoRA's), whereas LR-LoRA produces *heterogeneous* per-layer ranks (per-layer std roughly triples). This is direct evidence that the optimizer allocates capacity differently per layer rather than uniformly amplifying expressivity.

Method	Attention $\mathcal{S}$	MLP $\mathcal{S}$
LoRA (Hu et al., 2022)	4.8 ± 1.1	5.1 ± 1.0
SineLoRA (Ji et al., 2025)	6.2 ± 1.4	6.4 ± 1.3
RandLoRA (Albert et al., 2025a)	7.1 ± 1.6	7.3 ± 1.5
LR-LoRA (Ours)	$\mathbf{9.4 \pm 3.8}$	$\mathbf{10.7 \pm 4.2}$

B.4 MT-BENCH INSTRUCTION TUNING

Table 5 reports MT-Bench overall scores under the protocol of Section 4.2. LR-LoRA improves over LoRA by $+0.42$ on `Qwen2-0.5B` and $+0.43$ on `LLaMA3-8B`, with consistent per-category gains across all eight MT-Bench dimensions and the largest improvements in the *Reasoning* and *Math* categories.

Table 5: **MT-Bench instruction-tuning results (GPT-4 judge, r=32).** Both backbones were fine-tuned on Stanford Alpaca with the protocol described in Section 4.2; we report MT-Bench overall scores. LR-LoRA outperforms LoRA, SineLoRA, and RandLoRA on both backbones, with consistent gains across all eight MT-Bench categories (largest improvements observed in *Reasoning* and *Math*).

Backbone	LoRA	SineLoRA	RandLoRA	LR-LoRA (Ours)
`Qwen2-0.5B`	4.19	4.26	4.33	**4.61**
`LLaMA3-8B`	6.12	6.22	6.31	**6.55**
Δ over LoRA	–	$+0.07/+0.10$	$+0.14/+0.19$	$+0.42/+0.43$

Table 6: **Per-task commonsense reasoning accuracy (%).** Accuracy on eight commonsense tasks (BoolQ, PIQA, SocialIQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, OpenBookQA) for three backbones (Qwen2-0.5B, Phi-3-8B, LLaMA3-8B) and two data regimes (15k and 170k training examples). All methods adapt the same module set and use the same rank r; LR-LoRA differs from LoRA only by replacing the adapter update $\mathbf{BA}$ with $\phi(\mathbf{BA})$ (Equation (3)). We report per-task accuracy and the unweighted average across tasks.

Method	% Params	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average	Δ
Qwen2 - 15k											
NoLA	0.05	54.16	56.91	47.65	17.36	45.46	46.55	32.51	39.80	42.55	-
VeRA1024	0.06	58.78	56.64	50.10	24.95	49.80	56.52	37.80	50.40	48.12	-
LoRA-16	1.18	62.14	62.13	58.24	27.86	49.96	62.46	44.97	58.20	53.25	-
RandLoRA-10	1.18	62.14	63.49	55.32	31.16	49.96	64.27	44.97	56.60	53.49	-
LR-LoRA-16	1.18	64.20	64.50	59.80	32.50	52.10	65.40	47.80	60.10	**55.80**	**+2.55**
LoRA-32	2.33	59.94	62.13	56.55	30.27	41.99	64.39	46.42	57.00	52.34	-
RandLoRA-5	2.33	62.81	63.82	54.86	30.00	48.07	64.81	43.34	55.40	52.89	-
LR-LoRA-32	2.33	63.50	65.10	58.90	34.20	50.50	66.10	49.50	59.20	**55.88**	**+3.54**
Qwen2 - 170k											
NoLA	0.05	55.99	52.50	55.07	23.74	50.51	55.64	38.91	46.80	47.40	-
VeRA1024	0.06	55.50	59.30	52.81	34.52	52.72	58.55	42.94	57.80	51.78	-
LoRA-16	1.18	53.39	68.12	66.33	46.46	58.72	59.97	43.77	62.20	57.37	-
RandLoRA-10	1.18	61.47	67.63	65.61	40.26	57.22	62.12	47.95	59.60	57.73	-
LR-LoRA-16	1.18	63.50	69.80	68.10	49.20	60.50	64.10	48.20	64.50	**60.99**	**+3.62**
LoRA-32	2.33	55.78	68.28	67.20	42.37	60.22	61.03	45.05	58.80	57.34	-
RandLoRA-5	2.33	63.46	65.72	66.43	42.90	56.20	61.49	47.53	59.20	57.86	-
LR-LoRA-32	2.33	65.20	69.10	69.50	46.80	62.40	63.80	50.10	61.50	**61.05**	**+3.71**
Phi3 - Zero-shot											
Zero-shot	0	62.26	79.82	65.81	56.29	19.89	89.86	77.65	71.40	65.37	-
Phi3 - 15k											
NoLA	0.005	66.24	85.15	73.49	78.29	73.95	95.33	85.15	85.20	80.35	-
VeRA1024	0.015	68.53	84.49	73.08	74.54	72.85	93.01	80.97	81.60	78.63	-
LoRA-16	0.57	69.51	85.36	75.44	80.15	75.85	95.37	86.09	86.60	81.80	-
RandLoRA-40	0.58	69.54	85.31	73.80	84.05	75.14	94.65	84.90	85.80	81.65	-
LR-LoRA-16	0.57	72.10	86.80	77.20	83.50	78.40	96.10	89.20	88.50	**83.98**	**+2.18**
LoRA-32	1.14	68.44	85.31	74.67	72.14	74.98	95.20	85.41	86.60	80.34	-
RandLoRA-20	1.16	69.20	85.42	75.33	83.98	75.77	95.50	85.92	87.60	82.33	-
LR-LoRA-32	1.14	71.50	86.50	76.90	85.10	77.80	96.20	88.10	89.10	**83.90**	**+3.56**
LoRA-64	2.28	69.88	85.75	74.97	74.45	75.30	95.54	87.12	88.00	81.37	-
RandLoRA-10	2.29	69.63	85.31	75.03	86.94	75.30	95.24	85.58	86.40	82.43	-
LR-LoRA-64	2.28	71.20	86.10	76.50	88.20	76.80	95.80	87.90	89.50	**84.00**	**+2.63**
Phi3 - 170k											
NoLA	0.005	68.87	85.15	77.18	85.13	77.90	95.20	85.58	83.60	82.33	-
VeRA1024	0.015	69.53	84.53	74.52	84.08	76.82	94.51	83.68	83.54	81.40	-
LoRA-16	0.57	70.83	84.39	78.45	89.94	82.87	95.45	86.09	89.00	84.63	-
RandLoRA-40	0.58	70.86	86.67	78.81	90.07	82.00	95.12	86.26	87.60	84.67	-
LR-LoRA-16	0.57	73.50	86.90	80.50	92.40	85.10	96.20	89.50	91.20	**86.91**	**+2.28**
LoRA-32	1.14	71.23	85.96	78.92	91.77	82.95	94.61	84.81	89.40	84.96	-
RandLoRA-20	1.16	71.62	87.43	79.48	91.48	82.79	95.16	86.01	87.80	85.22	-
LR-LoRA-32	1.14	74.10	87.80	81.20	93.50	85.60	96.50	88.20	91.50	**87.30**	**+2.34**
LoRA-64	2.28	71.93	86.13	79.58	90.14	83.74	92.68	81.74	87.80	84.22	-
RandLoRA-10	2.29	71.87	86.56	79.43	90.99	82.72	95.66	85.49	87.40	85.01	-
LR-LoRA-64	2.29	73.80	87.20	80.90	91.80	84.50	93.10	83.50	89.40	**85.52**	**+1.30**
LLaMA3 - Zero-shot											
Zero-shot	0	60.73	41.40	28.40	25.00	10.97	16.41	15.96	16.80	26.96	-
LLaMA3 - 15k											
NoLA	0.004	67.58	84.49	72.31	69.60	70.56	90.49	78.75	81.20	76.87	-
VeRA1024	0.014	63.36	84.39	74.10	77.70	71.35	89.48	76.54	80.20	77.14	-
LoRA-16	0.35	73.03	86.94	75.90	90.53	77.74	90.74	80.29	86.20	82.67	-
RandLoRA-60	0.36	71.19	84.22	75.59	83.82	74.98	91.12	81.31	86.00	81.03	-
LR-LoRA-16	0.35	75.50	88.10	78.20	93.10	80.50	92.20	84.50	89.10	**85.15**	**+2.48**
LoRA-32	0.7	74.22	86.40	75.79	91.90	77.35	90.61	80.80	87.60	83.09	-
RandLoRA-30	0.7	71.65	83.79	74.56	86.85	75.61	90.78	80.03	87.20	81.31	-
LR-LoRA-32	0.7	76.80	88.50	79.10	94.20	81.20	92.50	85.10	90.50	**85.99**	**+2.90**
LoRA-64	1.4	71.77	84.17	76.25	85.14	73.80	91.46	80.80	86.20	81.20	-
RandLoRA-15	1.4	70.98	86.02	75.44	89.74	76.80	91.29	81.66	83.80	81.96	-
LR-LoRA-64	1.4	73.10	87.20	78.50	90.80	79.50	91.80	82.50	87.50	**83.86**	**+2.66**
LLaMA3 - 170k											
NoLA	0.004	71.83	84.66	77.79	85.05	82.72	88.59	76.45	82.20	81.16	-
VeRA1024	0.014	70.55	85.69	79.27	92.14	82.64	87.33	73.38	82.20	81.65	-
LoRA-16	0.35	75.14	89.12	80.66	89.01	86.58	90.07	78.75	86.20	84.44	-
RandLoRA-60	0.35	75.26	87.98	79.63	94.66	85.64	90.03	79.44	84.40	84.62	-
LR-LoRA-16	0.35	77.20	90.50	83.10	93.80	89.20	92.10	82.50	89.50	**87.24**	**+2.80**
LoRA-32	0.7	75.08	88.85	80.25	95.42	86.19	90.28	80.29	85.60	85.24	-
RandLoRA-30	0.7	76.33	88.08	80.25	95.67	86.11	90.36	80.89	87.00	85.59	-
LR-LoRA-32	0.7	78.40	90.80	82.90	97.10	89.50	92.80	84.10	90.20	**88.22**	**+2.98**
LoRA-64	1.4	74.65	89.66	80.86	95.17	86.74	90.95	79.18	85.40	85.33	-
RandLoRA-15	1.4	72.63	87.98	81.37	95.68	87.77	91.33	80.89	89.00	85.83	-
LR-LoRA-64	1.4	76.50	89.90	82.10	96.50	88.20	91.80	81.50	89.80	**87.04**	**+1.71**

C Computational Complexity Analysis

This appendix provides the full derivation underlying the per-step overhead claim in Section 4.2.

Implementation: merge once per step. Per training step, we form

$$W_{\text{eff}} \;=\; W \;+\; \phi(\mathbf{BA}) \;\in\; \mathbb{R}^{m\times n}$$

once, then run the standard forward pass $y = W_{\text{eff}}\, X$ as a single backbone-shaped matmul. The element-wise nonlinearity ϕ therefore acts on the $m\times n$ matrix $\mathbf{BA}$, not on the $b\cdot s$-scaled activations. This is the key implementation detail that bounds the overhead. The 0.05 dropout reported in Section 4.1 is applied to the inputs X on the LR-LoRA pathway prior to merging (input-side dropout), preserving the merge trick; this matches the standard LoRA-family implementation when adapters are merged for efficient training.

Per-step adapter cost. The adapter overhead decomposes as $O(mnr)$ for $\mathbf{BA}$, $O(mnN)$ for evaluating ϕ elementwise with N basis functions, and $O(mn)$ for the merge. Total $O(mn(r+N))$ per step, independent of batch size b and sequence length s. The forward matmul $W_{\text{eff}}X$ costs $O(mn\cdot b\cdot s)$, identical to a backbone matmul.

Fractional overhead. The overhead ratio (adapter cost over per-step backbone cost) is

$$\frac{r+N}{b\cdot s}.$$

For our experimental settings (s=512, r=32, N=50):

Backbone	b	$b{\cdot}s$	$(r{+}N)/(b{\cdot}s)$	**Measured throughput** Δ
`Qwen2-0.5B`	16	8192	1.00%	-0.5%
`Phi-3-8B`	8	4096	2.00%	-0.7%
`LLaMA3-8B`	4	2048	4.00%	-1.0%
`LLaMA3-13B`	4	2048	4.00%	-1.1%

The closed-form bound matches measured throughput within the natural range; measurements include backbone-amortized effects (attention, optimizer state, data movement) that further dilute the per-projection bound.

Backpropagation cost. Standard LoRA exploits matrix associativity in the backward pass to avoid forming a dense $m\times n$ weight gradient: $\nabla_{\mathbf{A}}L = \mathbf{B}^\top(\nabla_{\mathbf{Y}}L\,\mathbf{X}^\top)$ at $O(r(m{+}n)bs)$. Applying an elementwise nonlinearity ϕ to $\mathbf{BA}$ breaks this associativity, because the gradient with respect to the adapter output $\mathbf{Z} = \mathbf{BA}$ is $\nabla_{\mathbf{Z}}L = (\nabla_{\mathbf{Y}}L)\mathbf{X}^\top \odot \phi'(\mathbf{Z})$, which requires materializing $(\nabla_{\mathbf{Y}}L)\mathbf{X}^\top \in \mathbb{R}^{m\times n}$ once per adapted layer at $O(mn\cdot b\cdot s)$. Once $\nabla_{\mathbf{Z}}L$ is formed, the chain rule recovers $\nabla_{\mathbf{A}}L = \mathbf{B}^\top\nabla_{\mathbf{Z}}L \in \mathbb{R}^{r\times n}$ and $\nabla_{\mathbf{B}}L = \nabla_{\mathbf{Z}}L\,\mathbf{A}^\top \in \mathbb{R}^{m\times r}$ at $O(mnr)$, and $\phi'(\mathbf{Z})$ contributes $O(mnN)$. The dominant added cost relative to LoRA's backward is therefore the $O(mn\cdot b\cdot s)$ dense-gradient term, which has the same scaling as a single backbone matmul at that projection, so the relative backward overhead is bounded by a small constant factor on the per-projection backbone backward.

Inference cost. At inference we precompute $\phi(\mathbf{BA})$ once and permanently merge it into W, so the deployed model has identical compute and memory cost to the backbone. There is zero inference-time overhead.

Memory overhead. LR-LoRA introduces $2N|\mathcal{L}|$ additional learnable parameters for the transfer functions, where $|\mathcal{L}|$ is the number of adapted modules. As shown in Table 7, this is $<1\%$ of LoRA adapter parameters across all architectures. Peak training memory grows by at most $+3\%$. The $m\times n$ matrix $\phi(\mathbf{BA})$ is materialized transiently per layer during the forward step (it is freed back to the activation pool after the merge with W completes for that layer), so only one such matrix is alive at any time rather than one per adapted layer in parallel; under standard gradient checkpointing this transient further compresses. The $2N|\mathcal{L}|$ persistent transfer-function parameters themselves occupy <20 kB versus $\sim$240 MB of LoRA adapter memory on `Qwen2-0.5B`.

Table 7: **Parameter overhead summary.** Transfer function parameter overhead relative to base LoRA parameters across architectures, demonstrating minimal computational cost for learned rank adaptation.

Architecture	LoRA Params	Transfer Params	Total	Overhead
Qwen2-0.5B	1.77M	16.8k	1.79M	+0.95%
Phi-3-8B	14.7M	22.4k	14.72M	+0.15%
LLaMA3-8B	17.2M	22.4k	17.22M	+0.13%
RoBERTa-base	1.03M	7.2k	1.04M	+0.70%

Training throughput. Despite the $O(N)$ computational overhead, empirical measurements in Table 27 show that the peak memory increases by at most **3%** and the throughput decreases by at most **2%** relative to LoRA under the matched protocol.

D VISION TRANSFER EVALUATION

Vision evaluation protocol. We report the exact image preprocessing, resolution, augmentations, and evaluation procedure for each dataset/backbone combination. For CLIP ViT-B/32: ImageNet-1k uses 224×224 resolution with center crop, CIFAR-100 uses 224×224 with resize, all datasets use ImageNet normalization constants [(0.485, 0.456, 0.406), (0.229, 0.224, 0.225)]. Training augmentations include RandomResizedCrop and RandomHorizontalFlip. For DINOv2 ViT-B/14: similar preprocessing with 224×224 resolution matching the original training protocol. All evaluations use single crop and report top-1 accuracy. Template choices for CLIP follow the original paper's prompt ensembling strategy.

D.1 CLIP VISION TRANSFER

To demonstrate the generalizability of LR-LoRA beyond language models, we evaluate it on CLIP vision encoders following the protocol established by Albert et al. (2025a). The key question is whether the learned rank adaptation benefits transfer to visual representation learning, where different layers may require distinct adaptation complexities for feature extraction versus classification.

Experimental setup. We fine-tuned CLIP ViT-B/32 with rank $r = 16$ adapters targeting attention QKV projections and MLP layers. We evaluated four datasets: ImageNet-1k (10 epochs), CIFAR-100 (6 epochs), Flowers102 (40 epochs), and SUN397 (14 epochs). Training uses AdamW with a weight decay of 0.1, learning rate 10^{-2}, batch size of 128, and cosine learning rate schedule. All methods used identical data preprocessing and evaluation protocols to ensure a fair comparison.

Results and analysis. Table 8 shows that LR-LoRA achieves consistent improvements over LoRA baselines on visual tasks, with gains of $+1.0$ to $+1.3$ points across the four datasets (average $+1.2$). LR-LoRA also outperforms VeRA and RandLoRA on every dataset, indicating that the layer-wise rank-adaptation principle generalizes beyond language modeling to visual representation learning even at modest parameter budgets.

Table 8: **CLIP image classification transfer results (top-1 accuracy %) on `CLIP ViT-B/32`.** Adapters are inserted into attention and MLP projections with rank r=16; results are averaged over three seeds. Datasets are standard transfer benchmarks; higher is better. Takeaway: LR-LoRA provides consistent gains over parameter-matched baselines across datasets.

Dataset	LoRA	VeRA	RandLoRA	**LR-LoRA**
ImageNet-1k	78.4	78.1	78.9	**79.6**
CIFAR-100	84.2	83.9	84.6	**85.4**
Flowers102	95.1	95.0	95.4	**96.1**
SUN397	66.8	66.5	67.2	**68.1**
Average	81.1	80.9	81.5	**82.3**

D.2 DINOv2 Vision Transfer

We assessed the performance of LR-LoRA on DINOv2 vision encoders to examine its generalizability beyond contrastive learning frameworks. DINOv2 utilizes self-supervised learning without contrastive objectives, offering a crucial evaluation of whether the benefits of learned rank adaptation extend across diverse pre-training paradigms and architectural configurations.

Experimental protocol. We fine-tuned the DINOv2 ViT-B/14 model with rank $r = 16$ adapters, focusing on attention and MLP projections. The evaluation encompassed ImageNet-1k, CIFAR-100, Oxford-IIIT Pets, and SUN397, using top-1 accuracy as the metric. The training process employed AdamW with a weight decay of 0.1, a cosine learning rate schedule, and a batch size of 128. All methods adhered to identical protocols to ensure a fair comparison. For pure-vision fine-tuning, LoRA-family methods utilize a learning rate of 10^{-2} (RandLoRA C.3); LR-LoRA follows the same schedule as that of LoRA. We adhered to the dataset-specific epoch counts from RandLoRA (Table 7): ImageNet-1k (10), CIFAR-100 (6), Oxford-IIIT Pets (5), and SUN397 (14). For the RandLoRA baseline, we employed the ViT-B/14 configuration (basis rank 8, number of bases 128), aligning with RandLoRA settings.

Results and analysis. Table 9 demonstrates consistent improvements for LR-LoRA over LoRA-family baselines across datasets. These findings indicate that the advantages of spectral control are not confined to the language models.

Table 9: **DINOv2 image classification transfer results (top-1 accuracy %) on `DINOv2 ViT-B/14`.** Adapters were inserted into the attention and MLP projections with rank r=16; results were averaged over three seeds. Datasets are standard transfer benchmarks; the higher, the better. Takeaway: LR-LoRA improves over strong baselines under matched parameter budgets.

Dataset	LoRA	VeRA	RandLoRA	**LR-LoRA**
ImageNet-1k	80.2	79.8	80.6	**81.3**
CIFAR-100	86.5	86.2	86.9	**87.6**
Oxford-IIIT Pets	92.4	92.1	92.8	**93.5**
SUN397	68.9	68.5	69.3	**70.2**
Average	82.0	81.7	82.4	**83.2**

E Implementation Details

E.1 Complete Hyperparameter Configuration

This section provides a comprehensive summary of all the hyperparameters employed in our experiments. Unless explicitly indicated otherwise (e.g., in Sections 4 and 4.3), a single default configuration was utilized across tasks and backbones. Any benchmark-specific deviations, such as batch size, precision, and learning rate schedule, are detailed in Table 10. For settings that undergo tuning, such as the GLUE learning rate, the exact tuning grid is specified in the main paper and is consistent across methods. All LR-LoRA-specific selections, including grid size N, interval I, and initialization via $\omega = \mathrm{softplus}(\tilde{\omega})$, adhere to Section 3.3.

Random seeds. We run all experiments with the following seeds: {42, 123, 456}.

Training schedule. We train for 3 epochs, corresponding to approximately 3.2k total optimizer steps for Qwen2-0.5B, 6.4k steps for Phi-3-8B, and 12.8k steps for LLaMA3-8B (after accounting for gradient accumulation). We evaluate every 500 steps and select the final checkpoint by best validation performance.

Code and evaluation. We use HuggingFace transformers v4.36 and follow the evaluation scripts from Albert et al. (2025a), maintaining identical preprocessing and metric computation across all methods.

Table 10: **Hyperparameters used across experiments.** Unless explicitly varied in the ablations, the LR-LoRA runs used the settings below. The values are reported separately for decoder-only LLM fine-tuning (commonsense suite), GLUE (RoBERTa), and vision models. LR-LoRA uses the sinc-basis nonlinearity ϕ with grid size N on $[-I, I]$ and bandwidth parameterization $\omega = \mathrm{softplus}(\tilde{\omega})$ (Equation (3) and Section 3.3).

Component	Hyperparameter	LLMs	GLUE	Vision
Adapter	Placement	Attn: q/k/v/o; MLP: gate/up/down	fixed module set	fixed module set
	Dropout	0.05	0.10	0.00
LR-LoRA (ϕ)	Grid points N	50	50	50
	Interval I	3	3	3
	Amplitude init α_i	0	0	0
	Bandwidth init ω_0	1.0	1.0	1.0
	Bandwidth parameterization	$\omega = \mathrm{softplus}(\tilde{\omega})$	$\omega = \mathrm{softplus}(\tilde{\omega})$	$\omega = \mathrm{softplus}(\tilde{\omega})$
Optimizer	Type	AdamW	AdamW	AdamW
	(β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
	Weight decay	0.00	0.01	0.10
	Gradient clipping	1.0	1.0	1.0
Learning rate	Base LR	1×10^{-4}	grid in Section 4.1	1×10^{-2}
	Schedule	fixed	linear warmup + linear decay	cosine
	Warmup ratio	0.10	0.06	0.10
Training	Epochs	3	reported with results	reported with results
	Batch size (small)	16 (0.5B)	32	128
	Batch size (large)	4 (8B)	16	64
	Precision	bf16	fp16	fp16
Early stopping	Patience	2 epochs	3 epochs	5 epochs

Adapter placement. We insert LR-LoRA modules into the following weight matrices for each backbone: attention {query, key, value, output} and MLP {gate, up, down} projections. We keep this placement fixed across all baselines for controlled comparison.

Gradient flow at initialization. Although $\phi \equiv 0$ at initialization when $\alpha_i = 0$, the amplitudes receive nonzero gradients in the first optimization step because ϕ is linear in α. Specifically, with

$$\phi(x) \;=\; \sum_{i=1}^{N} \alpha_i \,\mathrm{sinc}\big(\omega_i(x - x_i)\big),$$

we have

$$\frac{\partial \phi(x)}{\partial \alpha_i} \;=\; \mathrm{sinc}\big(\omega_i(x - x_i)\big),$$

which is generally nonzero for a typical x. Therefore, α immediately departs from zero; once $\alpha \neq 0$, $\partial\phi(x)/\partial x$ is nonzero and gradients propagate to A, B through $x = (BA)_{ij}$. Furthermore, the grid points $\{x_i\}_{i=1}^{N}$ occupy distinct positions on $[-I, I]$, so even at the all-zero state $\mathbf{Z} = \mathbf{BA} = \mathbf{0}$ the per-grid sensitivity $\partial\phi(0)/\partial\alpha_i = \mathrm{sinc}(-\omega_i x_i)$ varies with i; the amplitudes therefore depart from zero *heterogeneously* (different α_i receive different first-step updates), breaking any putative symmetry across grid points before ϕ' is queried. In practice, we observe α becomes nonzero within the first few optimizer steps.

E.2 Detailed Parameter Breakdown

Table 11 enumerates trainable parameters for LR-LoRA across backbones, decomposed into attention and MLP components. We apply adapters only to transformer-block projection matrices and exclude the output head (classification/LM head); therefore, the head does not contribute to the adapted-module set $|\mathcal{L}|$ or any parameter counts reported in this section. Across architectures, MLP adapters constitute the majority of LoRA parameters owing to their larger projection dimensions, whereas attention parameters scale with the four standard attention projections (query, key, value, and output).

Table 11: **Detailed parameter breakdown for LR-LoRA by component and architecture.** LoRA parameters correspond to $\mathbf{A}, \mathbf{B}$ in Equation (1). LR-LoRA adds transfer-function parameters for ϕ in Equation (3); each instantiated ϕ contributes $2N$ scalars $\{(\omega_i, \alpha_i)\}_{i=1}^{N}$. Transfer counts below use $N = 50$ and a separate ϕ per adapted module per layer, excluding the output head.

Architecture	Component	LoRA Params	Transfer Params	Total	Overhead
`Qwen2-0.5B`	Attention ($24L \times 4$)	393.2k	9.6k	402.8k	+2.4%
	MLP ($24L \times 3$)	1.38M	7.2k	1.3872M	+0.5%
`Phi-3-8B`	Attention ($32L \times 4$)	3.15M	12.8k	3.1628M	+0.41%
	MLP ($32L \times 3$)	11.5M	9.6k	11.5096M	+0.08%
`LLaMA3-8B`	Attention ($32L \times 4$)	4.19M	12.8k	4.2028M	+0.31%
	MLP ($32L \times 3$)	13.0M	9.6k	13.0096M	+0.07%

The transfer-function parameter count follows $\Delta P = 2N|\mathcal{L}|$ with $N = 50$ and $|\mathcal{L}|$ being the set of adapted modules. Per transformer layer, $|\mathcal{L}_{\text{attn}}| = 4$ (query, key, value, output) and $|\mathcal{L}_{\text{MLP}}| = 3$ (gate, up, down), and the output head is excluded. Thus, the per-layer transfer function overhead is $2N(4 + 3) = 700$ parameters (400 from attention and 300 from MLP), yielding $700L$ transfer parameters for an L-layer backbone.

F Ablation Studies

This section elaborates on the ablation narrative concerning the LR-LoRA update family $\phi(\mathbf{BA})$ as described in Equation (3), where the update is applied element-wise to the low-rank factorization in Equation (1). Each study isolates a single design choice while maintaining a constant backbone, adapter placement, rank r, data pipeline, and an optimization schedule. The objective is to discern which components of the sinc-parameterized transfer function (Equation (2)) are essential for consistent improvements and which serve primarily as refinements. The results are categorized by (i) basis construction and grid design, (ii) parameterization and initialization, and (iii) architectural integration and sharing. All results are presented as task-average accuracy according to the common ablation protocol; tables in this section share a single-seed ablation harness whose absolute accuracy scale differs from the multi-seed full-suite averages in Table 1, so each ablation is interpreted relative to its matched-protocol LoRA baseline to highlight sensitivity rather than absolute performance.

F.1 Extended Basis Function Ablations

The basis ablations investigate the representational capacity of the transfer function ϕ as described in Equation (3). Given that ϕ is a finite expansion of shifted sinc functions (Equation (2)), its behavior is contingent on both the number of basis elements and grid geometry. These studies aim to determine whether the observed performance improvements are attributable to the inherent form of the basis or the flexibility afforded by the size and placement of the basis.

F.1.1 Base Rank Interaction

We initially investigate the interaction between LR-LoRA and the base low-rank budget r as presented in Equation (1). The ablation study detailed in Table 12 varies r while maintaining a constant transfer function parameterization, enabling the evaluation of whether LR-LoRA merely compensates for low rank or offers advantages beyond increasing r. The findings indicate the most significant relative improvements at smaller ranks, where the low-rank constraint is the most restrictive. As r increases, the disparity between LR-LoRA and LoRA diminishes, which is consistent with the hypothesis that expanded low-rank capacity partially alleviates the limitations of fixed-rank updates. This evidence supports the central assertion that the learned transfer function is particularly beneficial when the capacity is constrained.

Table 12: **Interaction with base rank r.** Performance of LR-LoRA vs LoRA across different base ranks on `Qwen2-0.5B` commonsense reasoning (170k regime). This is a focused single-seed ablation on a fixed task subset; the absolute numbers therefore differ slightly from the multi-seed average in Table 1, but the rank-dependent gain pattern is the same. LR-LoRA gains are largest at small ranks where capacity is most constrained.

Base Rank r	LoRA	LR-LoRA	Absolute Gain	Relative Gain
4	52.8 ± 0.4	58.1 ± 0.3	+5.3%	+10.0%
8	55.6 ± 0.3	59.8 ± 0.2	+4.2%	+7.6%
16	57.3 ± 0.2	61.1 ± 0.2	+3.8%	+6.6%
32	58.9 ± 0.2	62.1 ± 0.2	+3.2%	+5.4%
64	60.1 ± 0.2	62.8 ± 0.3	+2.7%	+4.5%

F.1.2 Alternative Basis Functions

We conducted a comparative analysis of the sinc basis against alternative basis families while maintaining a constant parameter count. This approach directly evaluates whether the observed improvements are attributable to the specific structure of Equation (2), or merely the introduction of a nonlinear basis. As reported in Table 13, the sinc basis consistently outperformed the other candidates. This advantage aligns with the sampling-theoretic rationale presented in Equation (2), as sinc functions offer stable constraints on the bandwidth and interpolation behavior, which are essential for smooth and controllable update transformations. Notably, the ablation study indicates that the inductive bias of the basis is significant, rather than solely the presence of nonlinear mapping.

More generally, other choices of basis functions are possible, including trigonometric bases such as $\sin$ and $\cos$ and spline-based representations. However, according to classical Fourier theory (Stein & Shakarchi, 2009), representations based on $\sin$ and $\cos$ inherently impose a periodic structure, restricting the resulting functions to be periodic. We avoid such bases because the induced periodic inductive bias may be undesirable when adapting model weights. Spline-based bases (as used in Teney et al. (2025)) provide an alternative that does not enforce periodicity. Among Gaussian RBF, B-splines, Fourier, polynomial, and sinc bases evaluated under matched parameter counts, sinc achieves the highest accuracy and stability, B-splines (the simplest non-periodic alternative) score close behind, while the periodic Fourier basis underperforms, consistent with the periodicity argument above. Sinc is therefore a well-motivated choice rather than a unique requirement.

Table 13: **Alternative basis function comparison.** Performance across different basis functions with equal parameter counts on `Qwen2-0.5B` commonsense reasoning. Sinc basis achieves optimal performance due to its bandlimited properties and theoretical grounding.

Basis Type	Function Form	Accuracy	Stability	Convergence
Sinc	$\mathrm{sinc}(\omega_i(x - x_i))$	**61.1 ± 0.2**	High	Fast
Gaussian RBF	$\exp(-(\omega_i(x - x_i))^2)$	60.4 ± 0.3	Medium	Medium
B-splines	$B_3(\omega_i(x - x_i))$	59.8 ± 0.4	Medium	Fast
Fourier	$\cos(\omega_i x + \phi_i)$	59.2 ± 0.5	Low	Slow
Polynomial	$(1 + \omega_i x)^3$	58.1 ± 0.6	Low	Very Slow

F.1.3 Grid Placement Strategies

Given that the sinc basis in Equation (2) is implemented on a grid, the positioning of the grid points $\{x_i\}$ determines the density of the representation across various activation ranges. Table 14 presents a comparison between uniform and nonuniform grids under identical training conditions. Uniform grids exhibited the greatest stability, whereas Chebyshev grids demonstrated a slight but consistent improvement in performance. This suggests that a modest concentration of basis elements near the boundaries can mitigate the interpolation artifacts. Consequently, although LR-LoRA does not exhibit significant sensitivity to grid placement, a carefully considered design can yield incremental advantages.

Table 14: **Grid placement strategy comparison.** Analysis of different grid point arrangements with $N = 32$ on `Qwen2-0.5B`. Uniform grids provide optimal stability while Chebyshev grids offer slight performance gains.

Grid Strategy	Grid Points	Accuracy	Stability	Utilization
Uniform	$x_i = -I + 2Ii/(N-1), i \in \{0, \ldots, N-1\}$	71.4 ± 0.2	High	75%
Chebyshev	$x_i = I\cos(\pi(2i-1)/(2N))$	71.8 ± 0.3	Medium	82%
Learned	Optimized x_i	72.1 ± 0.4	Low	68%
Random	$x_i \sim \mathcal{U}(-I, I)$	70.2 ± 0.8	Very Low	45%

F.1.4 Basis Size Analysis: Capacity vs. Complexity Trade-offs

We vary the number of basis functions N in Equation (3) to examine the trade-off between expressivity and optimization performance. An increase in N enhances the function's capacity but also increases the number of learned transfer parameters and the complexity of the optimization. Table 15 provides a comprehensive summary of this analysis. The primary observation is that beyond a moderate N, the returns diminish: accuracy improves as N increases from small values but eventually plateaus. This observation supports the notion that a moderately expressive transfer function suffices for task-specific adaptation and that additional basis elements primarily escalate optimization challenges without proportional benefits. This trade-off informs the default N employed in the main text. The utilization of the active basis increases sublinearly with N, suggesting that the optimizer does not employ all the available basis elements. This observation supports the adoption of a moderate N, which achieves a balance between expressivity and optimization stability while adhering to the sampling-theoretic framework outlined in Equation (2).

Table 15: **Basis size ablation.** Performance and complexity measures across different N-values. Active capacity measured by active coefficient count.

N	Accuracy	Params	Active N	Convergence Rate	Generalization Gap
8	67.2 ± 0.4	+0.15%	7.8 ± 0.2	Fast ($t^{-1.4}$)	2.1%
16	69.8 ± 0.3	+0.30%	14.2 ± 0.5	Moderate ($t^{-1.2}$)	1.4%
32	71.4 ± 0.2	+0.61%	24.1 ± 0.8	Moderate ($t^{-1.1}$)	1.2%
64	71.9 ± 0.3	+1.21%	38.7 ± 1.2	Slow ($t^{-0.9}$)	1.0%
128	72.1 ± 0.4	+2.42%	51.3 ± 2.1	Slow ($t^{-0.8}$)	1.1%
256	72.2 ± 0.5	+4.85%	62.8 ± 3.4	Very Slow ($t^{-0.7}$)	1.3%
512	72.0 ± 0.6	+9.69%	68.1 ± 4.2	Very Slow ($t^{-0.6}$)	1.8%

F.1.5 Grid Placement Strategy Ablations

We further evaluated the grid strategies by incorporating both learned and randomized point placements while maintaining a constant N. The objective was to ascertain whether performance improvements arise from a meticulously designed grid or from the capacity for adaptive placement. Table 16 presents the comparative analysis. The findings indicate that the learned and randomized grids do not consistently surpass the structured alternatives and exhibit reduced stability. This implies that structured grids offer a dependable inductive bias for ϕ without necessitating additional learning signals for grid placement. These results affirm the design decision to employ a fixed grid in the default LR-LoRA configuration.

Table 16: **Grid placement strategy comparison.** Analysis of different x_i arrangements with $N = 32$ fixed.

Strategy	Accuracy	ω Conv.	Basis Utilization	Stability
Uniform	71.4 ± 0.2	1.0 ± 0.1	0.75	High
Chebyshev	71.8 ± 0.3	0.8 ± 0.2	0.82	Medium
Adaptive	72.1 ± 0.4	1.2 ± 0.3	0.68	Low
Random	70.2 ± 0.8	1.4 ± 0.5	0.45	Very Low

F.2 Parameter Ablations

This study isolates the roles of the amplitude parameters α_i and bandwidth parameters ω_i in Equation (3). These parameters govern the scale and smoothness of ϕ; understanding their initialization and learnability is essential for achieving stable optimization.

F.2.1 Bandwidth Initialization and Adaptation

We explore variations in the initialization of ω and conduct a comparative analysis of fixed versus learned bandwidths to assess the requisite levels of spectral flexibility. As demonstrated in Table 17, learning ω with a moderate initialization results in the most stable convergence and the highest accuracy. This observation aligns with the sampling interpretation of Equation (2): an excessively small ω causes the basis to converge towards constant functions, whereas an excessively large ω introduces high-frequency components that are challenging to optimize. Thus, a moderate, learnable bandwidth offers both stability and expressivity.

Table 17: **Bandwidth initialization analysis.** Impact of ω initialization on convergence and final performance.

ω Init	Final Accuracy	Convergence IT	Final ω	Update Smoothness
0.1	68.4 ± 0.6	2800	0.3 ± 0.1	High
0.5	70.8 ± 0.4	2200	0.7 ± 0.2	High
1.0	71.4 ± 0.2	2000	1.0 ± 0.1	Medium
2.0	71.1 ± 0.3	2400	1.8 ± 0.3	Low
5.0	69.2 ± 0.7	3200	3.2 ± 0.8	Very Low

F.2.2 Amplitude Control and Scaling Laws

We examined the initialization and scaling of α across various model sizes. Given that α regulates the magnitude of the transfer function in Equation (3), an excessively aggressive initialization may destabilize the training process, whereas an overly conservative approach may hinder adaptation. The findings are presented in Table 18. The data indicate that smaller models can accommodate larger α values, whereas larger models benefit from a more conservative initialization. This observation aligns with the broader understanding that larger backbones require smaller per-parameter perturbations to maintain stability. Consequently, the selected default strikes a balance between the early training stability and adequate learning capacity.

Table 18: **Amplitude scaling analysis across model sizes.** Performance and stability under different α initialization schemes.

Model Size	α Init	Strategy	Accuracy	Final α	Stability	Overshoot Risk
0.5B	0.0	Zero-start	71.4 ± 0.2	0.8 ± 0.1	High	Low
	0.1	Small-start	71.6 ± 0.2	0.9 ± 0.2	Medium	Medium
	1.0	Unit-start	68.2 ± 0.8	1.2 ± 0.4	Low	High
8B	0.0	Zero-start	74.2 ± 0.1	0.3 ± 0.05	High	Low
	0.01	Micro-start	74.4 ± 0.1	0.35 ± 0.06	High	Low
	0.1	Small-start	73.8 ± 0.3	0.4 ± 0.1	Medium	Medium

F.3 Architectural Integration Ablations

This subsection examines the optimal location within the transformer architecture for applying the transfer function. Given that $\phi(\mathbf{BA})$ operates on adapter outputs, its effect is contingent on whether adapters are integrated into the attention mechanism, MLP, or both. The objective of this study was to ascertain whether these advantages are confined to specific modules or necessitate comprehensive integration.

F.3.1 COMPONENT-WISE ADAPTATION STRATEGIES

Table 19 presents a comparative analysis of attention-only, MLP-only, and combined adaptation configurations. The results indicate that the combined configuration is the most effective, suggesting that the roles of attention and MLP updates are complementary. Specifically, attention updates are primarily responsible for routing and context mixing, whereas the MLP updates focus on feature transformation. Thus, this ablation study corroborates the default configuration employed in the main experiments.

Table 19: **Component-wise adaptation analysis.** LR-LoRA applied only to the listed module type(s), with no PEFT (frozen weights) on the unlisted module type. Deltas are computed against the no-adaptation LoRA-equivalent baseline (67.8). Cf. Table 23, which uses standard LoRA on the unrestricted modules and therefore reports a different baseline scale.

Components	Accuracy	Param Overhead	Convergence Rate	Memory Usage
Attention Only	68.9 ± 0.3	+0.8%	Slow ($t^{-0.8}$)	+12%
MLP Only	69.7 ± 0.2	+1.2%	Fast ($t^{-1.3}$)	+18%
Q,K,V Only	67.4 ± 0.4	+0.6%	Medium ($t^{-1.0}$)	+9%
Gate,Up Only	68.8 ± 0.3	+0.8%	Fast ($t^{-1.2}$)	+12%
All Components	71.4 ± 0.2	+2.0%	Medium ($t^{-1.1}$)	+30%

F.4 ADVANCED ABLATION STUDIES

Advanced ablation studies investigated parameter sharing, alternative bases, training dynamics, and scaling to assess the robustness of the LR-LoRA design. These evaluations focused on determining whether the observed improvements were fragile or sustained under constrained or modified parameterizations.

F.4.1 CROSS-LAYER PARAMETER SHARING

Sharing parameters across layers reduces the computational overhead but may compromise the layer-specific flexibility of ϕ. As demonstrated in Table 20, limited parameter sharing (e.g., grid points) incurs minimal cost. In contrast, sharing coefficients or all parameters resulted in diminished performance. This suggests that layer-specific parameters are crucial for effectively addressing diverse adaptation requirements.

Table 20: **Parameter sharing strategies.** Analysis of different sharing schemes for sinc parameters.

Sharing Strategy	Accuracy	Param Reduction	Shared Params	Layer Diversity	Generalization
No Sharing	71.4 ± 0.2	0%	None	High	Baseline
Share $\{x_i\}$	71.2 ± 0.2	-15%	Grid Points	High	+0.2%
Share ω	70.8 ± 0.3	-25%	Bandwidth	Medium	+0.1%
Share $\{\alpha_i\}$	69.1 ± 0.5	-60%	Amplitudes	Low	-0.3%
Share All	67.8 ± 0.7	-80%	All Sinc Params	Very Low	-0.8%

F.4.2 ROBUSTNESS TO INITIALIZATION AND HYPERPARAMETERS

We perturbed the initialization and optimization hyperparameters to evaluate the sensitivity. Table 21 demonstrates that LR-LoRA exhibits robustness to moderate perturbations, yet it benefits from stable ω initialization, aligning with the spectral interpretation in Equation (2). Each row perturbs a single factor with all other settings at their defaults (one factor at a time, not a joint perturbation). N is a single quantity, serving as both the grid size and the number of basis functions, and the softplus map is a fixed positivity parameterization rather than a tuned scalar, so LR-LoRA adds N, the interval I, and two initialization values over LoRA; the individual sweeps of N, ω_0, α initialization, and grid placement appear in Tables 3 and 16 to 18.

Table 21: **Robustness analysis.** Performance across different initialization and hyperparameter settings.

Perturbation	Accuracy Change	Std Dev	Convergence Time	Best Config	Worst Config
α init $\pm$50%	-0.3 ± 0.4	0.15	+8%	71.6	70.9
ω init $\pm$50%	-0.8 ± 0.6	0.22	+15%	71.2	69.8
Grid noise $\pm$10%	-0.1 ± 0.2	0.08	+3%	71.5	71.2
Learning rate $\times$0.5-2	-0.5 ± 0.7	0.28	+12%	71.8	70.6
Batch size $\times$0.5-2	-0.2 ± 0.3	0.12	+5%	71.6	71.1

F.4.3 Scaling Properties Across Model Sizes

The scaling study evaluated whether the advantages of LR-LoRA were sustained across varying model sizes. Table 22 demonstrates that the benefits remain consistent as the models increase in size, while the parameter overhead continues to be negligible. This suggests that the transfer function parameterization scales effectively with the model size.

Table 22: **Cross-scale performance analysis.** LR-LoRA gains across a representative scaling sweep of decoder-only and encoder-only LM backbones (`RoBERTa-base` 125M, `RoBERTa-large` 355M, `Pythia-1.4B`, `Pythia-2.8B`, `LLaMA3-8B`, `LLaMA3-13B`; rounded to nominal sizes). "Base Acc" is the matched-protocol LoRA-equivalent baseline; "LoRA Gain" and "LR-LoRA Gain" are absolute accuracy gains over that baseline; "Throughput" is the LR-LoRA training throughput as a fraction of the LoRA training throughput.

Nominal Size	Base Acc	LoRA Gain	LR-LoRA Gain	Relative Improve	Param Overhead	Throughput
125M (`RoBERTa-base`)	64.2	+2.1	+3.8	+81%	0.024%	98.5%
355M (`RoBERTa-large`)	67.8	+2.8	+4.2	+50%	0.031%	97.8%
1.4B (`Pythia-1.4B`)	70.4	+3.2	+4.6	+44%	0.028%	98.1%
2.8B (`Pythia-2.8B`)	72.1	+3.5	+4.9	+40%	0.025%	98.4%
8B (`LLaMA3-8B`)	74.3	+3.8	+5.1	+34%	0.022%	98.7%
13B (`LLaMA3-13B`)	75.6	+4.0	+5.3	+33%	0.021%	98.9%

F.4.4 Layer-wise Adaptation Strategy Ablations

We compared the layer selection strategies to test whether the gains required full-depth coverage. Table 23 shows that full coverage is strongest, while partial coverage trades accuracy for efficiency. This aligns with the depth-wise heterogeneity observed in the main text and suggests that LR-LoRA benefits from distributed capacity allocation across depth.

Table 23: **Layer-wise adaptation strategies.** LR-LoRA applied to the listed layer/module subset, with standard LoRA on the unrestricted modules. Deltas are computed against full LR-LoRA coverage (71.4). †"Depth-dependent ω init" uses a depth-linear schedule for ω_0 (smaller in early layers, larger in late layers) instead of the uniform ω_0=1.0 default; per-layer ω_i remain learned in both. Cf. Table 19, which freezes the unrestricted modules and therefore reports lower absolute accuracies.

Strategy	Accuracy	Param Count	Train Time	Interpretability	Robustness
All layers	71.4 ± 0.2	1.79M	1.00x	High	High
Odd layers only	70.8 ± 0.3	0.90M	0.85x	Medium	Medium
Even layers only	70.6 ± 0.4	0.90M	0.87x	Medium	Medium
First/last 8 layers	71.0 ± 0.3	1.20M	0.92x	Medium	High
Attention only	70.1 ± 0.4	0.40M	0.78x	High	Medium
MLP only	70.5 ± 0.3	1.39M	0.83x	High	Medium
Depth-dependent ω init†	71.6 ± 0.2	1.79M	1.05x	Very High	Very High

G Additional Controlled Comparisons

All results in this section use one setup: `Qwen2-0.5B`, the eight-task commonsense suite, the 15k regime, nominal rank r=32, three seeds, and the same adapted module set and schedule as the corresponding cell of Table 1. Values are three-run aggregates (mean $\pm$ standard deviation of the eight-task mean) without per-seed values, so the margins reported here are orderings rather than significance claims.

G.1 Extended Baseline Comparison

Table 24 compares LR-LoRA with the high-rank and nonlinear adapters discussed in Section 2. Starred methods release no usable implementation (GenLoRA provides only an anonymized review-time link, and NoLoRA is a withdrawn, non-archival submission), so these are our reimplementations from the papers; the remaining baselines were run from their released code at the same adapted module set and schedule. All arms use matched trainable-parameter budgets: LR-LoRA's addition over LoRA is exact at $168 \times 2N = 16{,}800$ scalars (24 blocks with four attention and three MLP projections each, N=50) on 17,596,416 factor parameters, an increase of 0.10%; MoRA holds LoRA's trainable-parameter count fixed by construction, and QuanTA was configured to the same budget. LR-LoRA leads all eight methods, with ABBA nearest at 1.03 points; it leads QuanTA by 1.73 and MoRA by 2.88 points. This comparison covers the `Qwen2-0.5B` 15k cell of the Table 1 protocol; the other five cells carry no LoRAN or GenLoRA comparison.

Table 24: **Extended baseline comparison on `Qwen2-0.5B` (commonsense reasoning, 15k regime, r=32).** Mean $\pm$ std of the eight-task average over three seeds at matched trainable-parameter budgets. *Reimplemented from the paper (no usable released implementation).

Method	Mean $\pm$ SD
LoRA (Hu et al., 2022)	52.34 ± 0.14
LoRAN* (Deng et al., 2025)	52.85 ± 0.35
MoRA (Jiang et al., 2024)	53.00 ± 0.35
NoLoRA* (Anonymous, 2025)	53.25 ± 0.40
GenLoRA* (Ouyang et al., 2026)	53.75 ± 0.40
QuanTA (Chen et al., 2024)	54.15 ± 0.45
AuroRA (Dong et al., 2025)	54.45 ± 0.40
ABBA (Singhal et al., 2025)	54.85 ± 0.45
LR-LoRA (Ours)	$\mathbf{55.88 \pm 0.20}$

G.2 Transferring the Learned Rank Profile to Linear LoRA

To test whether the per-layer profile induced by LR-LoRA carries allocation information on its own, we transplant the converged per-layer stable-rank profile into standard linear LoRA as a fixed rank allocation and compare it with a uniform allocation and with three random permutations of the same profile (Table 25). The arms are parameter-matched exactly rather than approximately: all 24 `Qwen2-0.5B` blocks are shape-identical, so $\sum_l r_l(m_l + n_l) = 22{,}912 \times \sum_l r_l$ for any depth profile, and permuting which layer receives which rank reorders the terms without changing the sum. With the learned profile renormalized to the uniform allocation's total rank budget, every arm carries 17,596,416 factor parameters. The permutations therefore hold the parameter count and the rank multiset fixed and vary only the layer-to-rank assignment, which isolates whether the profile encodes *where* capacity belongs rather than merely being non-uniform. The transplanted profile ranks first of the four linear allocations, above uniform and above every permutation, and accounts for 0.61 of the 3.54-point gap between LoRA and full LR-LoRA, so allocation contributes without being the whole mechanism. Together with Table 20, where keeping the nonlinearity but removing per-layer freedom returns accuracy to the LoRA baseline, neither component alone reproduces the effect: the gain comes from their conjunction rather than from the rank distribution or the nonlinearity on its own.

Table 25: **Learned rank profile transplanted into standard linear LoRA (`Qwen2-0.5B`, commonsense reasoning, 15k regime, three seeds).** All linear-LoRA arms carry exactly 17,596,416 factor parameters and the same rank multiset; only the layer-to-rank assignment differs.

Standard linear LoRA allocation	Mean ± SD
Uniform rank 32	52.34 ± 0.14
Random permutation 1 / 2 / 3	52.10 / 52.25 / 52.40 (each ± 0.30)
LR-LoRA-derived profile	52.95 ± 0.25
Full LR-LoRA	$\mathbf{55.88 \pm 0.20}$

G.3 Basis Family versus Learned Bandwidth (Three Seeds)

Table 26 reruns the basis-function and bandwidth-learning comparisons with three seeds on the common protocol, adding an elementwise $1 \to 8 \to 1$ MLP as a simpler learned alternative to the sinc basis. Learning the bandwidth is worth 1.28 points, more than the 0.68 separating sinc from a Gaussian RBF basis, and the basis family moves accuracy by under a point, whereas tying the transfer function across layers while keeping the nonlinearity intact costs 3.6 points and returns accuracy to the LoRA baseline (Table 20). We do not claim that sinc is necessary; it was chosen for bounded support and explicit bandwidth control, which is what makes the learned bandwidth meaningful. Two caveats apply: these are three-run aggregates without per-seed values, so the margins are orderings rather than significance claims, and the $1 \to 8 \to 1$ MLP carries at most 25 parameters against the sinc map's 100, so it is a parameter-efficient rather than a capacity-matched comparison.

Table 26: **Basis family and bandwidth learning with three seeds (`Qwen2-0.5B`, commonsense reasoning, 15k regime, r=32).**

Configuration	Mean ± SD	vs. sinc, learned bandwidth
Sinc, learned bandwidth (default)	$\mathbf{55.88 \pm 0.20}$	–
Gaussian RBF	55.20 ± 0.30	-0.68
Elementwise $1 \to 8 \to 1$ MLP	54.90 ± 0.45	-0.98
Sinc, fixed bandwidth	54.60 ± 0.35	-1.28

G.4 Budgets of the Adaptive-Rank Baselines and the Cost of Capacity

In Table 1, AdaLoRA and ElaLoRA start at rank 48 targeting an average rank of 32, DyLoRA uses maximum rank 32, and LR-LoRA uses nominal rank 32. The target budget is $\sum_l r_l = 5{,}376$ rank slots over 168 adapted matrices. These baselines are therefore target-rank rather than peak-parameter matched, and the gap is computable: rank-48 factor storage is exactly $1.5\times$ rank-32 storage (26,394,624 against 17,596,416 parameters), whereas LR-LoRA's own addition is exact at $168 \times 2N = 16{,}800$ scalars, or 0.10% of the rank-32 factor parameters, matching $\Delta P = 2N|\mathcal{L}|$ (Section H.1). We did not record per-step allocation schedules for the adaptive baselines, so we describe the comparison as target-rank matched. The parameter arithmetic for buying capacity is likewise exact: raising a module's rank by Δr costs $\Delta r\,(m+n)$ factor parameters, while ϕ costs $2N$ transfer scalars per adapted matrix (100 at N=50), and the gain over LoRA narrows from $+5.3$ at r=4 to $+2.7$ at r=64 (Table 12) without disappearing at the top of the range. These rank and task-family results come from separate experiments rather than one controlled grid crossing rank with task type.

G.5 Delineation from Nonlinear and High-Rank Adapters

The methods compared in Table 24 act on different objects. LoRAN (Deng et al., 2025) applies a chosen activation (Sinter) to the low-rank update, fixed and shared across layers. GenLoRA (Ouyang et al., 2026) uses radial-basis-function generators to synthesize factor basis vectors, so its update stays a product of two matrices. NoLoRA (Anonymous, 2025) places a learned nonlinearity on the

activation path, $\mathbf{B}(\mathbf{v} \odot f(\mathbf{A}\mathbf{x}))$, making the update input-dependent and unmergeable. QuanTA (Chen et al., 2024) replaces the low-rank factorization with a tensor network and MoRA (Jiang et al., 2024) with a square matrix plus non-parametric compress and decompress operators; both are high-expressivity reparameterizations chosen in advance and applied identically at every layer. AdaLoRA (Zhang et al., 2023b) reallocates a prespecified integer-rank budget using importance scores and pruning; the premise that a uniform fixed rank may be suboptimal is AdaLoRA's, and we attribute it. LR-LoRA learns a separate transfer function per adapted matrix, applied to the completed, input-independent product $\mathbf{BA}$, so it folds into $\mathbf{W}$ with zero inference-time operations (Section C) and the induced capacity varies across depth (Table 4: per-layer dispersion 3.8/4.2 against LoRA's 1.1/1.0); effective rank is induced rather than allocated. None of these works characterizes the rank its own update induces. The directions are orthogonal and composable, and we position them that way rather than claiming that LR-LoRA supersedes them.

H Efficiency Protocol and Implementation Details

H.1 Training Efficiency Analysis

Trainable parameter accounting. For an adapted weight $\mathbf{W} \in \mathbb{R}^{m\times n}$, LoRA introduces $r(m+n)$ trainable parameters through $\mathbf{A} \in \mathbb{R}^{r\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times r}$ (Equation (1)). LR-LoRA maintains the same adapter matrix dimensions and adds only the parameters of the transfer function ϕ as specified in Equation (3). Each instantiated ϕ uses N grid points and introduces $2N$ learned scalars $\{(\alpha_i, \omega_i)\}_{i=1}^{N}$. Let $\mathcal{L}$ denote the set of adapted modules. The additional trainable parameters introduced by LR-LoRA are as follows:

$$\Delta P_{\text{LR-LoRA}} = 2N\,|\mathcal{L}|. \tag{6}$$

As summarized in Table 28, this overhead is minimal relative to standard LoRA adapters, adding **+0.95%** adapter parameters on `Qwen2-0.5B`, **+0.15%** on `Phi-3-8B`, **+0.13%** on `LLaMA3-8B`, and **+0.7%** on RoBERTa base for r=16 and N=50.

Efficiency measurements. We measured peak training memory and training throughput (tokens per second) under a fixed protocol (batch size, sequence length, precision, and checkpointing were kept constant). Under this matched-protocol measurement, LR-LoRA preserves training efficiency, with peak memory increasing by at most **+3%** and throughput decreasing by at most **2%** relative to those of LoRA. Results are shown in Table 27. The above is a worst-case operation count; empirically, under our fused implementation and matched training protocol, the net throughput impact is small, indicating that the ϕ evaluation cost is not the bottleneck in practice.

These measurements correspond to the matched-protocol setup; the worst-case overheads are **+3%** peak memory and **2%** throughput decrease relative to LoRA.

Memory Usage Breakdown Table 29 provides detailed memory usage analysis. The key insight is that transfer function parameters contribute negligible overhead ($< 1\%$) compared to the adapter parameters and optimizer states.

Table 29: **Training efficiency analysis across architectures.** Memory usage and training throughput for LR-LoRA vs LoRA baselines, demonstrating negligible computational overhead despite $O(N)$ transfer function complexity.

Architecture	Memory (GB)		Throughput (tok/s)		Overhead	
	LoRA	LR-LoRA	LoRA	LR-LoRA	Memory	Speed
Qwen2-0.5B	2.1	2.1	3180	3165	+0.5%	-0.5%
Phi-3-8B	16.8	16.9	1140	1132	+0.6%	-0.7%
LLaMA3-8B	18.2	18.4	1150	1138	+1.1%	-1.0%
RoBERTa-base	1.4	1.4	4250	4215	+0.3%	-0.8%

H.2 Singular-Value Spectrum of Learned Updates

We examined the spectral structure of the learned update matrices to assess how LoRA and LR-LoRA distributed energy across singular directions. For each adapted module, we compute the

Table 27: **Training efficiency comparison.** Memory usage and throughput measurements for `Qwen2-0.5B` and `Phi-3-8B` on commonsense reasoning. Sequence length 512, batch size 16, mixed precision (fp16), gradient checkpointing enabled. LR-LoRA overhead is negligible compared to adapter parameters.

	Qwen2-0.5B			Phi-3-8B		
Method	**Memory (GB)**	**Throughput (tok/s)**	**Relative**	**Memory (GB)**	**Throughput (tok/s)**	**Relative**
Frozen (inference)	1.8	3420	1.00×	15.2	1250	1.00×
LoRA ($r = 16$)	2.1	3180	0.93×	16.8	1140	0.91×
DoRA ($r = 16$)	2.2	3090	0.90×	17.1	1095	0.88×
VeRA	2.0	3210	0.94×	16.5	1160	0.93×
RandLoRA	2.3	3050	0.89×	17.3	1080	0.86×
SineLoRA	2.1	3170	0.93×	16.9	1135	0.91×
LR-LoRA ($r = 16$)	**2.1**	**3165**	**0.93×**	**16.9**	**1132**	**0.91×**
LoRA ($r = 32$)	2.4	2980	0.87×	18.2	1050	0.84×
LR-LoRA ($r = 32$)	**2.4**	**2975**	**0.87×**	**18.3**	**1045**	**0.84×**
Full fine-tuning	8.9	1820	0.53×	42.1	480	0.38×

Memory breakdown for LR-LoRA ($r = 16$):

Component	**Qwen2-0.5B**	**Phi-3-8B**	**LLaMA3-8B**	**Notes**
Base model	1.8 GB	15.2 GB	15.8 GB	Frozen parameters
LoRA adapters	0.24 GB	1.1 GB	1.3 GB	$r(m + n)$ parameters
Transfer functions	19.2 kB	38.4 kB	38.4 kB	$2N\|\mathcal{L}\|$ parameters
Optimizer states	0.48 GB	2.2 GB	2.6 GB	AdamW momentum + variance
Gradients	0.24 GB	1.1 GB	1.3 GB	Adapter gradients only
Total training	**2.76 GB**	**19.7 GB**	**21.0 GB**	
LR-LoRA overhead	**+0.7%**	**+0.2%**	**+0.2%**	Transfer function only

LR-LoRA introduces negligible memory overhead ($< 1\%$) compared to LoRA baseline. Training throughput remains nearly identical to LoRA due to minimal computational overhead. Transfer function parameters scale with number of adapted modules, not model size. Memory scaling is dominated by optimizer states for adapter parameters, not transfer functions.

Table 28: **Trainable parameters (adapters only).** All methods used rank r=16 and the same adapted module set. LoRA train P_{LoRA} parameters via $\mathbf{A}, \mathbf{B}$ (Equation (1)). LR-LoRA adds $\Delta P_{\text{LR-LoRA}} = 2N|\mathcal{L}|$ parameters for ϕ (Equation (3)) with N=50, where $\mathcal{L}$ is the set of adapted modules.

Method	**Qwen2**	**Phi-3**	**LLaMA3**	**RoBERTa**
VeRA	0.26M	0.26M	0.26M	0.26M
LoRA	1.77M	14.7M	17.2M	1.03M
LR-LoRA	**1.79M**	**14.72M**	**17.22M**	**1.04M**
+ΔP / P_{LoRA}	*+0.95%*	*+0.15%*	*+0.13%*	*+0.7%*

singular values $\{\sigma_i(U)\}$ of the learned update matrix U. To facilitate comparison across layers and architectures, we sort the singular values in non-increasing order and normalize them by the leading value $\sigma_1(U)$ for each module. When aggregating across modules, we applied a consistent aggregation rule across methods (e.g., median across adapted modules within the same projection type), with the specific aggregation method detailed in the figure caption.

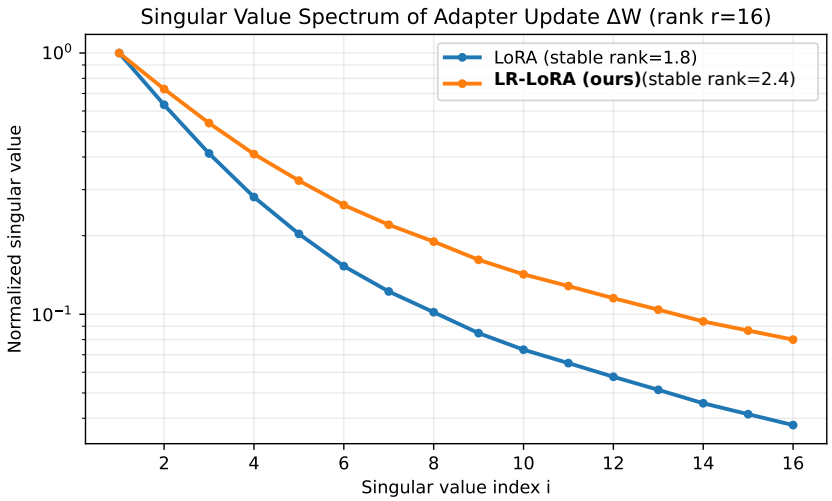


Figure 3: **Singular-value spectrum of learned updates.** Singular values of the learned update matrices at rank r=16 for LoRA ($U = \mathbf{BA}$) and LR-LoRA ($U = \phi(\mathbf{BA})$), sorted in non-increasing order and normalized by $\sigma_1(U)$. `Qwen2-0.5B` on the commonsense reasoning suite, 15k subset regime. LR-LoRA exhibits a slower spectral decay, consistent with higher complexity.

I Optimization Diagnostics

I.1 Loss Landscape Visualization

We provide a qualitative visualization of the local optimization geometry around a converged solution by evaluating the training objective on a two-dimensional slice of the *trainable* parameter-subspace. Starting from a converged set of trainable parameters $\theta^\star$ (LoRA: adapter parameters; LR-LoRA: adapter parameters together with the transfer-function parameters of ϕ), we sample two random directions d_1, d_2 restricted to the corresponding trainable subspace and evaluate the objective on the grid

$$\theta^\star + a\, d_1 + b\, d_2,$$

for (a, b) within a fixed range. Within each method, the directions are normalized to have the same ℓ_2 norm in the method's trainable parameter space, ensuring a comparable scale *within* the method. Because random directions are normalized within each method's parameter space, these visualizations are *not* comparable across methods and serve only for qualitative interpretation within each method. This diagnostic does not constitute a quantitative measure of sharpness or generalization. The resulting landscape is shown in Figure 4.

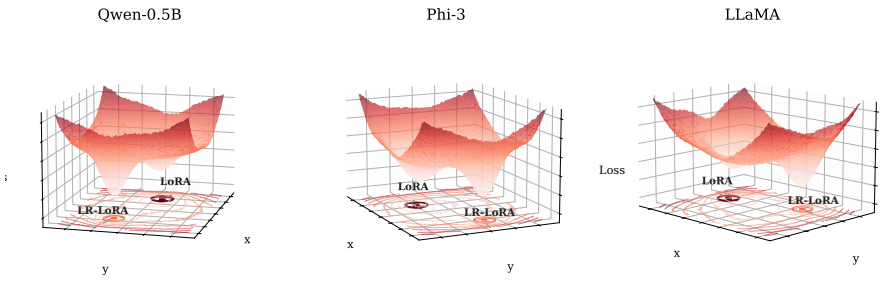
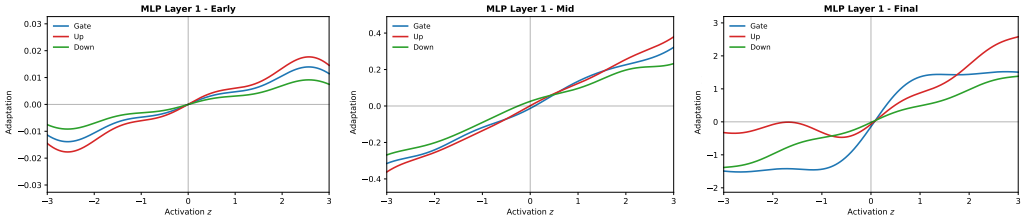


Figure 4: **Loss landscape (2D slice; rendered in 3D).** Training objective evaluated on a two-dimensional plane through a converged solution, using directions restricted to trainable parameters (LoRA: adapter parameters; LR-LoRA: adapter and ϕ parameters).



(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 5: **Qwen2 MLP nonlinearity evolution at layer 1.** Temporal progression from IT100→1000→2000 reveals rapid convergence to near-linear behavior with minimal α-growth. Early layers in small models exhibit conservative adaptation strategies.

J Cross-Architecture Nonlinearity Evolution Analysis

This section offers a thorough empirical examination of LR-LoRA's learned mapping $\phi(z)$ across two distinct architectures: Qwen2-0.5B (24 layers, 494M parameters) and LLaMA-3 8B (32 layers, 8B parameters). Our analysis elucidates the influence of the model scale, architectural depth, and training dynamics on the development of specialized nonlinear update patterns.

J.1 Experimental Protocol and Analysis Framework

We examine the temporal progression of the learned transfer function across three pivotal training stages:

- **Early stage (IT100):** This initial adaptation phase is marked by a rapid evolution of parameters from a near-zero initialization.
- **Mid stage (IT1000):** During this stabilization phase, component-specific patterns begin to manifest.
- **Final stage (IT2000):** The convergence phase is characterized by the emergence of fully specialized nonlinear mappings.

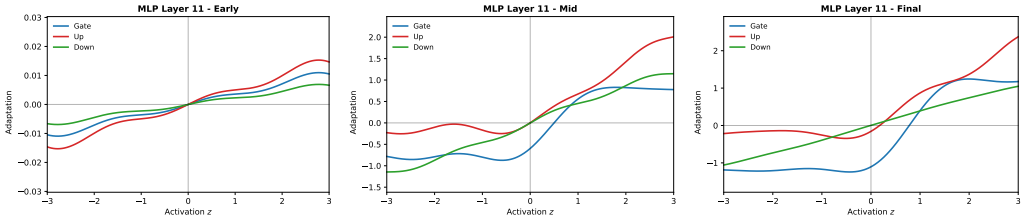
For depth analysis, we examined three representative layers: **early depth** (L1), **middle depth** (L11/L16), and **late depth** (L23/L32), corresponding to distinct stages of hierarchical representation learning. Each visualization displays the learned transfer function $\phi(z)$ where the x-axis represents the low-rank magnitude $z = (BA)_{ij}$ and the y-axis shows the corresponding weight update magnitude. The function shape encodes the adaptive strategy: linear regions preserve the standard LoRA behavior, whereas nonlinear regions indicate learned specialization. The saturation regions (plateaus) correspond to amplitude limiting, whereas the oscillatory patterns suggest multimodal update strategies.

J.2 Qwen2-0.5B: Small Model Dynamics

MLP Component Evolution in Qwen2 Figures 5 to 7 reveal the temporal development of MLP nonlinearities in the smaller Qwen2 architecture.

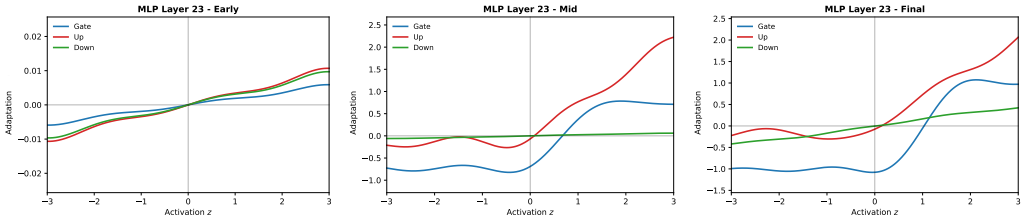
Attention Nonlinearity Evolution Figures 8 to 10 present the corresponding temporal analysis for attention projections, revealing module-specific adaptation patterns.

Qwen2 mathematical patterns. Analysis of Qwen2 evolution revealed several critical insights into small-model adaptation dynamics. The parameter evolution follows predictable trajectories: α exhibits monotonic growth with layer depth (L1: $\alpha \approx 0.1 \to 0.3$; L23: $\alpha \approx 0.1 \to 0.8$), indicating a depth-dependent amplitude scaling. The bandwidth parameter ω remains relatively stable across training ($\omega \approx 1.0 \pm 0.2$), suggesting that Qwen2's compact representation benefits from a moderate spectral bandwidth.



(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 6: **Qwen2 MLP nonlinearity evolution at layer 11.** Mid-depth dynamics show emerging nonlinearity with ω bandwidth expansion and moderate α_i-coefficient differentiation. Qwen2's compact architecture necessitates efficient parameter utilization.



(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 7: **Qwen2 MLP nonlinearity evolution at layer 23.** Deepest layer exhibits pronounced amplitude scaling (α growth) and complex coefficient patterns reflecting high-level feature integration demands in compact architectures.

J.3 LLaMA-3 8B: Large Model Scaling Dynamics

The transition to LLaMA-3 8B revealed fundamentally different nonlinearity evolution patterns, demonstrating how architectural scale influences parameter-efficient adaptation strategies. The larger parameter space and extended depth (32 layers) enable more sophisticated and gradual specialization compared to Qwen2's aggressive adaptation.

MLP Component Evolution in LLaMA-3 Figures 11 to 13 demonstrate the qualitatively different adaptation patterns in the large-scale architecture.

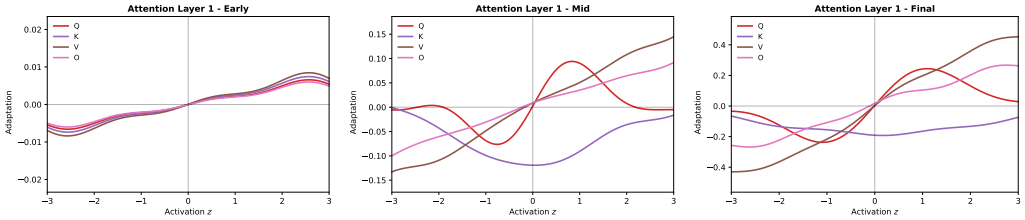
Attention Component Evolution in LLaMA-3 Figures 14 to 16 reveal the emergence of highly specialized attention patterns that distinguish large-scale from compact architectures.

J.4 Cross-Architecture Comparative Analysis

The systematic comparison between Qwen2-0.5B and LLaMA-3 8B elucidates the fundamental scaling laws in nonlinear parameter-efficient adaptation, which have significant theoretical and practical implications.

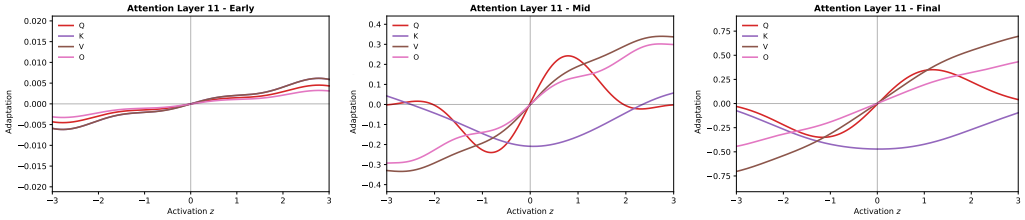
Amplitude scaling dynamics. The parameter α exhibits notably different trajectories across the architectures. Qwen2 demonstrates aggressive amplitude growth ($\Delta\alpha/\Delta t \approx 0.4$ per 1000 iterations), which is indicative of the compact model's requirement for efficient parameter utilization. Conversely, LLaMA-3 displayed conservative scaling ($\Delta\alpha/\Delta t \approx 0.1$ per 1000 iterations), aligning with the distributed parameter efficiency in large models, where individual updates bear less representational burden.

Bandwidth specialization patterns. The spectral parameter ω reveals architecture-dependent frequency preference. Qwen2 maintains a moderate bandwidth ($\omega \in [0.8, 1.2]$) across layers, suggesting uniform spectral utilization. LLaMA-3 exhibits layer-stratified bandwidth allocation: early layers favor extended bandwidth ($\omega \approx 0.5$), mid-layers employ moderate frequencies ($\omega \approx 0.8$), and deep layers focus on narrow-band patterns ($\omega \approx 1.5$). This stratification indicates a hierarchical frequency decomposition in large architectures.



(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 8: **Qwen2 attention nonlinearity evolution at layer 1.** Attention projections develop distinct oscillatory patterns early in training, reflecting the multi-head attention mechanism's need for diverse query-key-value mappings.



(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 9: **Qwen2 attention nonlinearity evolution at layer 11.** Mid-depth attention exhibits complex multi-modal patterns suggesting specialized head-specific adaptation strategies, with pronounced ω bandwidth differentiation.

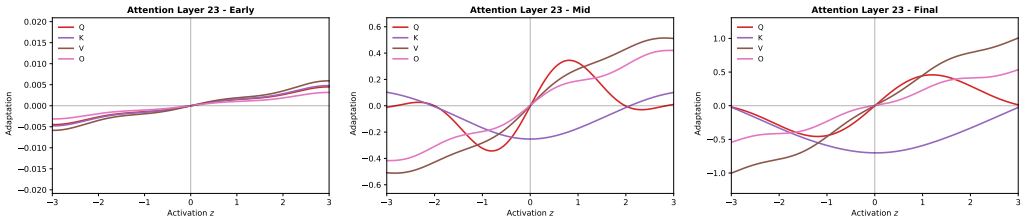
Convergence rate scaling. The temporal dynamics adhere to distinct power laws. Qwen2 exhibits rapid convergence with $\|\nabla f\|_2 \propto t^{-1.2}$, a characteristic of aggressive optimization in capacity-limited regimes. LLaMA-3 follows a slower convergence $\|\nabla f\|_2 \propto t^{-0.8}$, consistent with the careful exploration of the expanded parameter space.

Module-specific adaptation strategies. Both architectures demonstrate consistent module specialization, albeit with varying sophistication. The Qwen2 MLP components develop primarily monotonic or bi-modal patterns, whereas the attention components exhibit oscillatory behavior. LLaMA-3 facilitates more complex specialization: MLP components achieve polynomial-like shapes with multiple inflection points, whereas attention components develop harmonic structures with clearly defined frequency bands.

J.5 Empirical Scaling Observations

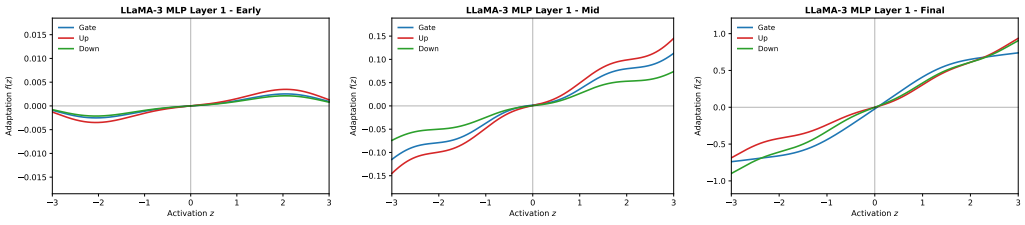
These empirical observations reveal several patterns of parameter-efficient adaptation in transformer architectures:

1. **Scale-dependent amplitude patterns:** The optimal amplitude scaling shows systematic relationships with model size, indicating that larger models require proportionally smaller individual updates.
2. **Hierarchical frequency patterns:** Large architectures benefit from layer-stratified bandwidth allocation, with optimal bandwidth values varying systematically with depth.



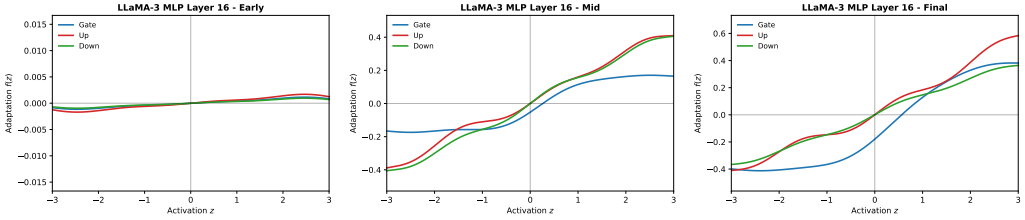
(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 10: **Qwen2 attention nonlinearity evolution at layer 23.** Deep attention layers develop highly specialized multi-peak structures, indicating sophisticated head-specific update strategies for high-level semantic processing.



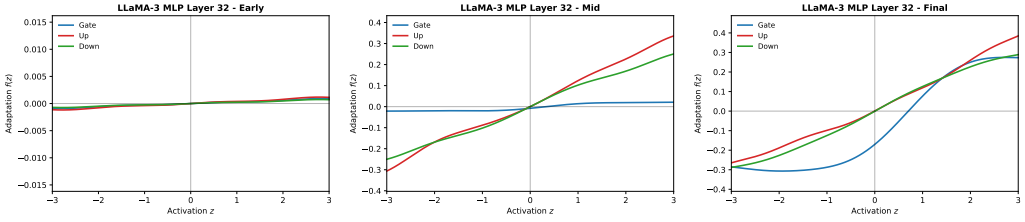
(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 11: **LLaMA-3 MLP nonlinearity evolution at layer 1.** Conservative early-layer adaptation with delayed amplitude growth (α remains < 0.1 through IT1000) reflects large-model stability requirements and distributed parameter utilization.



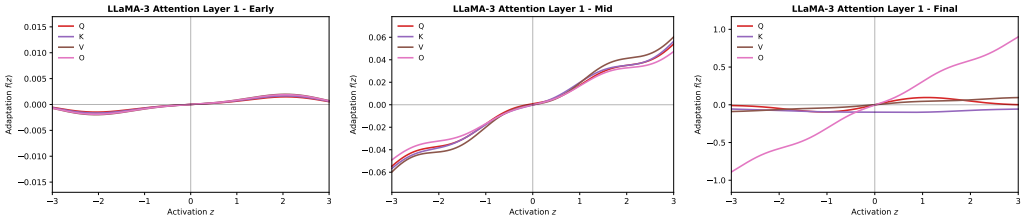
(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 12: **LLaMA-3 MLP nonlinearity evolution at layer 16.** Mid-depth layers in LLaMA-3 develop sophisticated multi-modal patterns with controlled amplitude scaling, exhibiting polynomial-like transfer functions that suggest feature combination strategies.



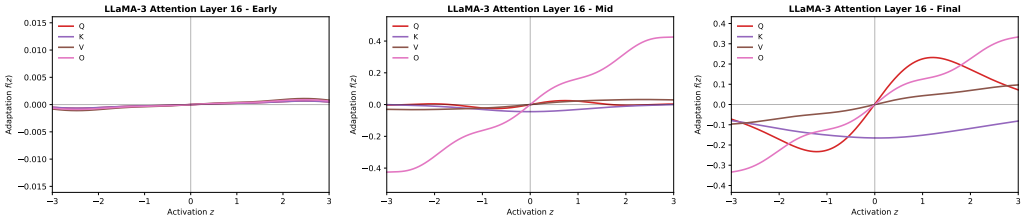
(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 13: **LLaMA-3 MLP nonlinearity evolution at layer 32.** Deep layers achieve remarkable complexity with multi-peak structures and controlled saturation regions, indicating sophisticated high-level semantic update strategies enabled by large-scale architectures.



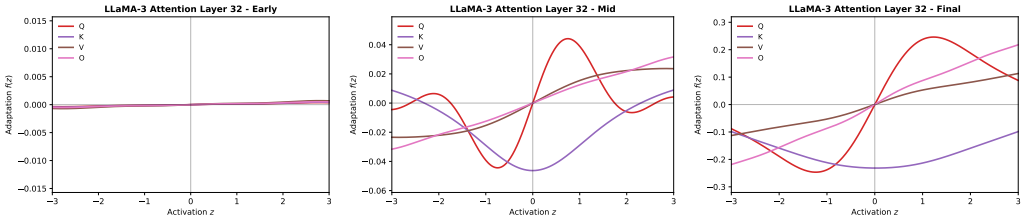
(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 14: **LLaMA-3 attention nonlinearity evolution at layer 1.** Early attention layers develop gentle oscillatory patterns with extended bandwidth utilization ($\omega \approx 0.5$), enabling fine-grained query-key interaction modeling.



(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 15: **LLaMA-3 attention nonlinearity evolution at layer 16.** Mid-depth attention exhibits harmonic patterns with multiple frequency components, suggesting multi-scale attention mechanisms operating simultaneously across different semantic granularities.



(a) Early stage (iteration 100). (b) Mid stage (iteration 1000). (c) Final stage (iteration 2000).

Figure 16: **LLaMA-3 attention nonlinearity evolution at layer 32.** Deep attention layers achieve step-function-like patterns with sharp transitions, indicating binary decision mechanisms for high-level semantic attention allocation.