CURVATURE-GUIDED RANK SCHEDULING FOR LORA FINE-TUNING OF VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation (LoRA) makes transfer learning parameter-efficient, but its rank is usually fixed before optimization and shared across layers. We introduce Curvature-Guided Rank Scheduling (CGRS), which uses a diagonal empirical-Fisher signal to change LoRA capacity during training. A global scheduler grows or shrinks one shared rank, while layer-wise schedulers allocate heterogeneous ranks across Vision Transformer (ViT) blocks; within each block, the query and value projections share a common rank. Rank changes preserve the learned update through a singular value decomposition (SVD)-based transfer and are delayed by a grace period to avoid reacting to noisy early curvature. We evaluate ImageNet-pretrained ViT-Base/16 on CIFAR-100, Street View House Numbers (SVHN), and Oxford Flowers-102 against full fine-tuning, a frozen backbone, a 14-rank LoRA sweep, and AdaLoRA. Global CGRS reaches 91.16% on CIFAR-100 and 97.45% on SVHN, improving over fixed LoRA at the same final rank by 0.70 and 0.47 percentage points, respectively. It also exceeds the best tested AdaLoRA accuracy by 1.10 and 1.54 points on these datasets. On Flowers-102, however, AdaLoRA is better (98.78% versus 98.52%), and some layer-wise schedules collapse to the maximum rank. These results support curvature as a useful capacity signal while identifying low-data calibration and one-way rank growth as important failure modes. Given single-seed evaluation, we treat the results as a pilot study establishing feasibility rather than a statistically validated comparison; multi-seed replication is the immediate next step (Section 7).

1 INTRODUCTION

Fine-tuning a pretrained model independently for every downstream task duplicates nearly all model parameters. Parameter-efficient fine-tuning (PEFT) instead learns a small task-specific state while keeping the backbone frozen. LoRA (Hu et al., 2022) represents the update to a pretrained weight as a low-rank product, making the trainable and stored state proportional to a chosen rank r. This benefit comes with a consequential hyperparameter: r must be selected before training and is commonly shared across modules, although downstream tasks and transformer depths need not demand equal adaptation capacity.

Adaptive-rank methods remove part of this manual search. Adaptive Budget Allocation for PEFT (AdaLoRA) (Zhang et al., 2023b) begins with an over-parameterized singular value decomposition (SVD)-like update and prunes low-importance singular triplets; Dynamic Search-Free LoRA (DyLoRA) (Valipour et al., 2023) trains nested representations usable at multiple ranks; and Incremental Parameter Allocation for PEFT (IncreLoRA) (Zhang et al., 2023a) grows modules according to importance. These methods establish that rank should be treated as a training-time variable. We investigate a complementary signal: curvature estimated directly from squared task gradients. Intuitively, persistent sensitivity of the loss to an adapter indicates that its current subspace may be too restrictive, whereas uniformly small sensitivity provides little evidence that more rank is needed.

We propose Curvature-Guided Rank Scheduling (CGRS), a family of schedulers for LoRA fine-tuning. CGRS probes the diagonal empirical Fisher, summarizes it globally or per transformer block, and compares it with calibrated thresholds. The global scheduler can both grow and shrink along a discrete rank ladder. Layer-wise schedulers assign one rank to each block, shared by its query 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.

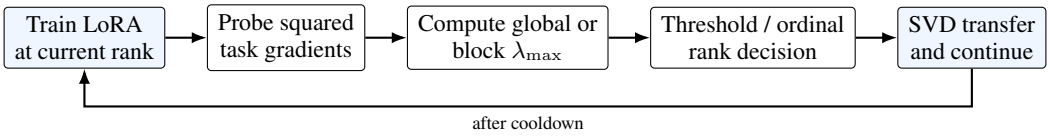

Figure 1: CGRS alternates ordinary LoRA optimization with curvature probes. After a grace period, the scheduler changes rank only at check points; SVD-based transfer preserves the learned update across the resize.

value adapters, and isolate optimizer resets to resized blocks. SVD-based transfer carries the learned low-rank update across each resize. Figure 1 summarizes the procedure.

Our contributions are:

- an empirical characterization of when and why curvature-guided rank adaptation succeeds and fails in vision transfer, including a documented threshold-calibration failure mode in the low-data Flowers-102 regime;
- a task-level global scheduler with bidirectional rank changes, SVD-based transfer, and explicit grace-period and cooldown controls, together with layer-wise variants that expose allocation behavior across transformer depth; and
- a controlled ViT-Base/16 study spanning three vision benchmarks, 14 fixed LoRA ranks, two conventional fine-tuning baselines, and three AdaLoRA budgets.

2 Related Work

Parameter-efficient adaptation. LoRA freezes a pretrained matrix W_0 and learns a factorized update BA, reducing task-specific storage and optimizer state (Hu et al., 2022). Although first developed for language models, the construction applies to transformer projections generally, including Vision Transformers (ViTs) (Dosovitskiy et al., 2021). Standard LoRA assigns ranks manually; our study asks whether a task-gradient signal can allocate rank online in vision transfer.

Adaptive rank. AdaLoRA parameterizes updates with singular triplets, scores their importance, and progressively prunes to a target budget (Zhang et al., 2023b). DyLoRA optimizes an ordered, nested range of ranks to avoid a separate search for each rank (Valipour et al., 2023). IncreLoRA grows parameter budgets incrementally instead of pruning from a large initialization (Zhang et al., 2023a). Like IncreLoRA, CGRS can grow from a smaller model; unlike these methods, its allocation statistic is an empirical-Fisher curvature proxy, and its global form also supports shrinking.

Concurrent curvature-aware PEFT. Recent concurrent preprints overlap more directly with our use of curvature. Layer-Aware Adaptive Rank Allocation (LAARA) dynamically assigns layer-wise ranks using normalized diagonal-Fisher estimates and a vote-to-change mechanism (Tripathi et al., 2026). GRIT, a geometry-aware PEFT method, combines Kronecker-factored approximate curvature (K-FAC) preconditioning, Fisher-guided basis reprojection, and spectrum-driven effective-rank adaptation (Saha et al., 2026). Curvature-Guided LoRA (CG-LoRA) instead uses curvature-whitened gradients to select and scale adaptation directions for prediction alignment (Zheng & Proutière, 2026). Consequently, we do not claim the first use of curvature or Fisher information for adaptive PEFT. Our narrower distinction is a vision-transfer study of a single task-level rank that can grow and shrink globally, SVD-based transfer across rank changes, explicit grace-period and cooldown stabilization, and a documented calibration failure on Flowers-102. LAARA emphasizes relative layer-wise allocation, while GRIT and CG-LoRA study geometry-aware basis or direction selection on language tasks.

Curvature signals. The Fisher information connects local model sensitivity and second-order optimization geometry (Martens, 2020). Full Fisher matrices are impractical at ViT scale, so we use the diagonal empirical approximation obtained from squared gradients. We do not claim that this proxy recovers the full Hessian; it is used only as a consistent, inexpensive control signal for capacity decisions.

3 CURVATURE-GUIDED RANK SCHEDULING

3.1 LORA ADAPTATION

For a frozen projection $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA computes

$$h = W_0 x + s_r B A x, \qquad A \in \mathbb{R}^{r \times d_{\text{in}}},\ B \in \mathbb{R}^{d_{\text{out}} \times r}, \tag{1}$$

where $s_r = \alpha / r$. We adapt the query and value projections in each of the 12 ViT-Base blocks, giving 24 adapters. With $d_{\text{in}} = d_{\text{out}} = 768$, a uniform rank uses $24(768 + 768)r = 36{,}864r$ adapter parameters.

3.2 DIAGONAL EMPIRICAL-FISHER SCORE

Let θ denote the LoRA parameters and $\mathcal{P}$ a probe set of mini-batches. For probe batch b, let $g_b = \nabla_\theta \ell_b(\theta)$. We estimate the diagonal empirical Fisher as

$$\widehat{f} = \frac{1}{|\mathcal{P}|} \sum_{b \in \mathcal{P}} g_b \odot g_b. \tag{2}$$

For adapter m, the scheduler uses

$$\lambda_m = \max_{i \in m} \widehat{f}_i, \qquad T_m = \sum_{i \in m} \widehat{f}_i. \tag{3}$$

The global statistic takes the maximum over all trainable LoRA coordinates; layer-wise scheduling uses the maximum query/value curvature within each transformer block to make one shared rank decision for that block. We track the trace T diagnostically but drive rank decisions with λ.

3.3 GLOBAL SCHEDULER

Before adaptive runs, we profile fixed-rank checkpoints and collect the 24 module scores at $r = 16$. Their 25th, 50th, and 75th percentiles define aggressive, moderate, and conservative thresholds τ. Global CGRS begins at $r = 16$ and selects ranks from

$$\mathcal{R} = \{3, 5, 6, 10, 12, 16, 30, 52, 64\}. \tag{4}$$

No transition is allowed during the first 15% of training. At a check point after the cooldown, $\lambda > \tau$ moves upward on $\mathcal{R}$ (with larger overshoot permitted to skip rungs), while $\lambda < 0.3\tau$ moves down one rung. Otherwise rank is unchanged. The global optimizer is recreated after a resize, and its learning rate is scaled by the old-to-new rank ratio.

3.4 LAYER-WISE SCHEDULERS

Layer-wise CGRS starts every transformer block at $r = 30$ and restricts ranks to $[3, 64]$; the query and value adapters within a block share that rank. Resizing a block resets only the Adam state of its two adapters, preserving momentum elsewhere. We evaluate four allocation policies:

- **PL-C1**: every block is compared with the shared static median threshold;
- **PL-C3-Live**: each block receives its own threshold from an $r = 30$ warm-up (200 steps or the corresponding fraction on short runs), multiplied by 0.5;
- **Ordinal-K4**: at each check, rank transformer blocks by curvature and grow the top four, while maintaining rank at least 16 for implementation-labeled blocks 4–7; and
- **Ordinal-K5**: the same block-level ordinal rule, growing the top five blocks.

PL-C1 and PL-C3-Live are implementation identifiers for the shared-static and per-block live-calibrated policies, respectively; they are not additional mathematical quantities. In Ordinal-K4 and Ordinal-K5, K denotes the number of blocks selected for growth at each check. The choice of blocks 4–7 is a hand-specified mid-depth safeguard intended to prevent starvation under repeated top-k selection; it is not a set inferred by CGRS from the Phase 2 curvature profile. These policies only grow; unlike the global scheduler, they cannot recover from transient over-allocation. This asymmetry becomes important in Section 6.

3.5 SVD-BASED RANK TRANSFER

Changing tensor shapes without transferring the learned update would restart adaptation. For current factors, we form $BA = U\Sigma V^\top$ and use the leading singular subspace to populate the resized factors. When shrinking, this is the best rank-r' approximation in Frobenius norm; when growing, it embeds the existing update into the larger factorization before training continues. The transition therefore retains the learned function up to truncation and the rank-dependent LoRA scaling.

4 EXPERIMENTAL SETUP

Backbone and datasets. All experiments use ImageNet-pretrained Vision Transformer Base with 16×16 patches (ViT-Base/16) (Dosovitskiy et al., 2021). CIFAR-100 (Krizhevsky, 2009) is split into 45,000/5,000/10,000 train/validation/test images. The Street View House Numbers (SVHN) (Netzer et al., 2011) training set is split into 68,000/5,257 images, with the 26,032-image test set unchanged. For Oxford Flowers-102 (Nilsback & Zisserman, 2008), the official train and validation sets are combined and re-split into 1,840/200 images; the official 6,149-image test set remains untouched.

Optimization. We train CIFAR-100 and SVHN for 3 epochs (8,439 and 12,750 steps) and Flowers-102 for 20 epochs (2,300 steps). All runs use batch size 16, AdamW (Loshchilov & Hutter, 2019), weight decay 0.01, cosine learning-rate decay, and gradient clipping at norm 1.0. Learning rates are 5×10^{-5} for full fine-tuning, 10^{-3} for a frozen backbone with a trained classifier, and 5×10^{-4} for LoRA-family methods. LoRA uses $\alpha = 16$ and dropout 0.1. The random seed is fixed to 42 for Python, NumPy, and PyTorch, including the Compute Unified Device Architecture (CUDA) backend.

Curvature profiling and layer-wise checks use 16 probe batches; global adaptive runs use 8–16 depending on the dataset. CIFAR-100 and SVHN use a 200-step check interval and 400-step cooldown. Because Flowers-102 has only 2,300 steps, its final protocol uses 2.4% and 4.7% of training steps (approximately 55 and 108 steps), respectively. All final adaptive runs gate transitions behind the 15% grace period.

Baselines and reporting. We compare full fine-tuning, classifier-only training, and fixed LoRA at $r \in \{3, 5, 6, 10, 12, 16, 30, 52, 64, 80, 100, 128, 150, 200\}$. AdaLoRA uses the PEFT implementation with target ranks 16, 32, and 64 and initial rank 1.5 times the target. Its warm-up/final phase each spans 15% of training and allocation is updated every 2.4% of steps. Accuracy is measured on the held-out test set. For LoRA-family methods, parameter counts denote final adapter parameters and exclude the dataset-specific classifier shared within a dataset. AdaLoRA's stored parameterization reflects its over-parameterized initialization. Given single-seed evaluation, we treat these results as a pilot study establishing feasibility rather than a statistically validated comparison; multi-seed replication is the immediate next step (Section 7).

5 RESULTS

5.1 GLOBAL RANK ADAPTATION

Table 1 compares the most accurate global CGRS variant on each dataset with fixed LoRA at the same final rank and with the most accurate tested AdaLoRA configuration. On CIFAR-100, conservative CGRS reaches 91.16%, 0.70 points above fixed $r = 64$ with the same 2.36M final adapter parameters. It also exceeds AdaLoRA's 90.06% while using 33% fewer adapter parameters.

On SVHN, conservative CGRS ends at $r = 52$ and reaches 97.45%. The exact capacity-matched baseline is fixed $r = 52$ at 96.98%, so the improvement is 0.47 points. Relative to fixed $r = 64$, the gain is 0.62 points with 19% fewer final adapter parameters. CGRS also improves over the best AdaLoRA accuracy by 1.54 points, although that AdaLoRA run uses fewer parameters (0.89M versus 1.92M); thus this is an accuracy improvement, not a Pareto dominance claim.

Flowers-102 reverses the pattern: aggressive global CGRS reaches 98.52%, below both fixed $r = 64$ (98.73%) and AdaLoRA target $r = 32$ (98.78%). This result localizes the weakness to adaptation rather than the backbone: classifier-only training reaches 99.20% on this dataset.

Table 1: Best global CGRS run per dataset. "Fixed" uses the same rank as the final CGRS checkpoint. AdaLoRA is selected independently by test accuracy, so its budget need not match. Parameters are final/stored adapter parameters in millions.

Legend: τ is the calibration percentile; accuracies and Δ are percentage points; P is final/stored adapter parameters in millions; Ada denotes AdaLoRA.

Dataset	τ	Final r	CGRS	Fixed	Δ	CGRS P	Best Ada (P)
CIFAR-100	75th	64	**91.16**	90.46	+0.70	2.36	90.06 (3.54)
SVHN	75th	52	**97.45**	96.98	+0.47	1.92	95.91 (0.89)
Flowers-102	25th	64	98.52	98.73	−0.21	2.36	**98.78** (1.77)

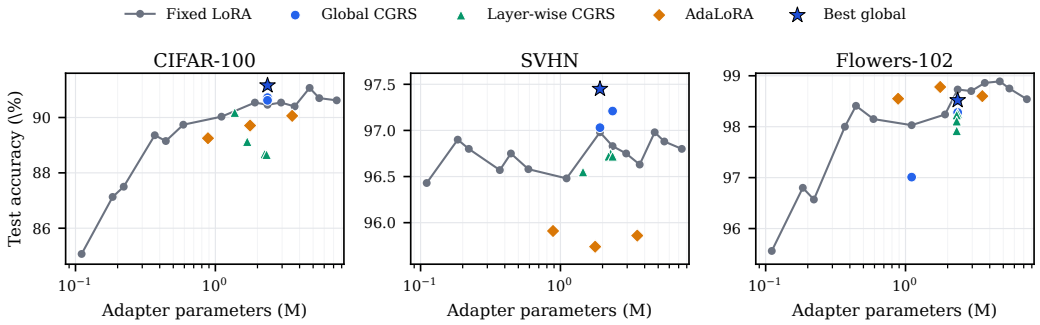

Figure 2: Test accuracy versus adapter parameters. Gray curves show the 14 fixed-rank LoRA runs; colored markers show adaptive methods. Stars denote the best global CGRS point in Table 1. Counts describe adapter storage, not total backbone size or peak training memory.

Figure 3 confirms that the adaptive runs are not fixed-rank models in disguise. The strongest CIFAR-100 run expands through several ranks, and the strongest SVHN run finishes at 52 after later adjustments. Flowers-102 exhibits substantially more late-stage grow–shrink oscillation, consistent with poorly calibrated curvature thresholds in the low-data regime.

5.2 Layer-wise allocation

Table 2 reports heterogeneous layer-wise schedulers. PL-C3-Live is the strongest layer-wise configuration on CIFAR-100: 90.18% at 1.38M parameters, within 0.28 points of fixed $r = 64$ while using 42% fewer final adapter parameters. On SVHN, PL-C1 allocates the least capacity (average rank 39.33) and reaches 96.55%; the task's fixed-rank curve is nearly flat, with 96.43–96.98% across ranks 3–200. This weak dependence on rank helps explain why low allocations remain competitive.

Layer-wise methods are less effective on Flowers-102. Average ranks converge to 63–64, and PL-C3-Live becomes exactly uniform at rank 64. Ordinal-K5 likewise becomes uniform on SVHN. These outcomes are allocation failures, not evidence that every block independently needs the maximum rank.

Figure 4 makes the allocation patterns explicit. SVHN PL-C1 and CIFAR-100 PL-C3-Live concentrate capacity in early blocks, whereas Flowers-102 PL-C3-Live assigns every block the maximum rank, exposing its calibration failure.

6 Analysis and Discussion

Does curvature influence allocation? We compare each final block-shared rank vector, expanded across the 24 query/value adapters, with two references: static $r = 16$ curvature measured during profiling and live $r = 30$ curvature measured before rank changes. Pearson tests linear association and Spearman tests ordering. The clearest result is SVHN PL-C1, whose final ranks correlate

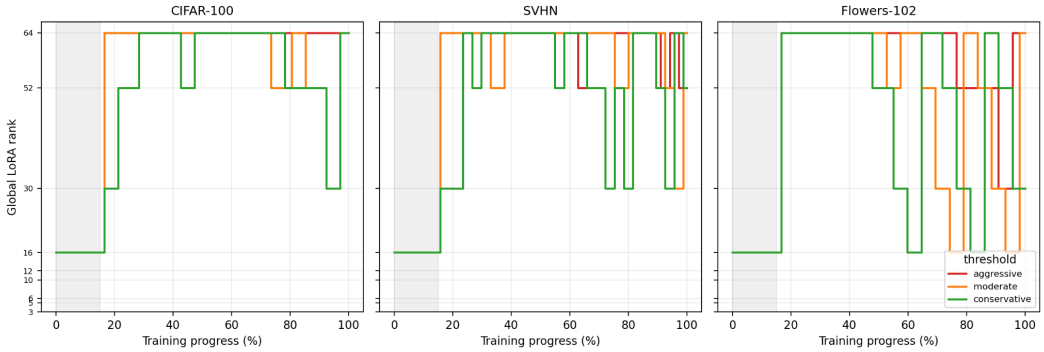

Figure 3: Global CGRS rank trajectories across training progress. Gray bands indicate the 15% grace period; colored curves show rank schedules under aggressive, moderate, and conservative curvature thresholds across CIFAR-100, SVHN, and Flowers-102.

Table 2: Layer-wise CGRS results. Average rank is over the 12 transformer blocks (equivalently, over 24 query/value adapters with a shared within-block rank); parameters are final adapter parameters. "Uniform" flags a constant final rank vector, for which rank–curvature correlation is undefined.

Legend: Avg. r is the mean final block rank; Params (M) is final adapter parameters in millions; Acc. is test accuracy in percent; Status indicates whether the 12-block final rank vector is heterogeneous or uniform.

Dataset	Scheduler	Avg. r	Params (M)	Acc.	Status
CIFAR-100	PL-C1	45.83	1.690	89.13	heterogeneous
	PL-C3-Live	37.33	1.376	**90.18**	heterogeneous
	Ordinal-K4	61.00	2.249	88.68	heterogeneous
	Ordinal-K5	63.00	2.322	88.66	heterogeneous
SVHN	PL-C1	39.33	1.450	96.55	heterogeneous
	PL-C3-Live	62.00	2.286	**96.75**	heterogeneous
	Ordinal-K4	60.00	2.212	96.72	heterogeneous
	Ordinal-K5	64.00	2.359	96.72	uniform
Flowers-102	PL-C1	63.00	2.322	97.92	heterogeneous
	PL-C3-Live	64.00	2.359	**98.26**	uniform
	Ordinal-K4	63.00	2.322	98.23	heterogeneous
	Ordinal-K5	63.00	2.322	98.11	heterogeneous

positively with static curvature (Pearson $r = 0.4325$, $p = 0.0348$). Other non-uniform runs are generally positive but do not reach $p < 0.05$; uniform runs make correlation undefined. Because query/value adapters within a block share a rank decision, these adapter-level correlations contain paired decisions and should be interpreted descriptively. Together with the single-seed design, the evidence is suggestive rather than definitive.

Why can a trajectory beat its final fixed rank? Fixed and adaptive runs can end with the same parameterization while following different optimization paths. CGRS changes the subspace dimension and resets some optimizer state; therefore its advantage at equal final rank should be interpreted as a schedule effect, not as a superior architecture. Establishing which transition—capacity growth, SVD transfer, or optimizer restart—causes the gain requires ablations not available in the current experiments.

Low-data instability. On Flowers-102, curvature values are approximately 10^{-5}–10^{-4}, and the global scheduler repeatedly grows and shrinks (for example, $16 \to 64 \to 52 \to 30 \to 16 \to 64$ in one conservative run). Layer-wise methods have no shrink branch, so one early threshold crossing permanently increases capacity. The halved live threshold in PL-C3-Live is especially permissive and

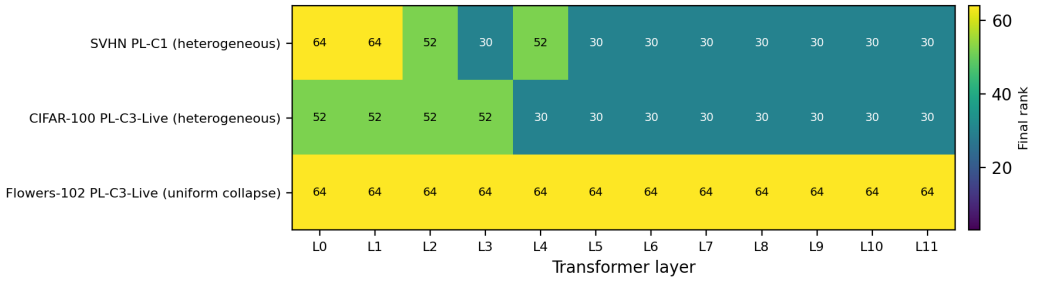

Figure 4: Representative per-layer final-rank allocations. CGRS produces heterogeneous layer-wise ranks on SVHN and CIFAR-100, while Flowers-102 PL-C3-Live collapses to the uniform maximum rank under threshold miscalibration.

drives every block to rank 64. The negative Flowers result is therefore consistent with a calibration failure, although the present experiments do not establish a causal mechanism.

Engineering safeguards matter. Three corrections were necessary before obtaining the reported results. First, rank decisions were blocked for the first 15% of steps; without this gate, Flowers adapters began growing within the first few checks. Second, cadence was expressed fractionally on the short Flowers run so it received a meaningful number of adaptation opportunities. Third, Flowers training was extended from 3 to 20 epochs: the 3-epoch LoRA runs had only about 345 updates and remained at 66–70%, whereas the corrected sweep reaches 95.56–98.89%. These are protocol corrections, not post-hoc exclusions of individual outcomes; all Flowers tables use the corrected setting.

7 Limitations

The most important limitation is statistical: every configuration has one seed, so 0.5–1.5 point differences may include run-to-run noise. The study uses one backbone and classification only, leaving language, detection, larger ViTs, and modern multimodal models untested. Thresholds are calibrated separately from an $r = 16$ checkpoint and may inherit dataset-specific scale; Flowers demonstrates that a fixed multiplier is not robust. Layer-wise scheduling only grows and can permanently over-allocate. Final adapter counts do not measure peak training memory, wall-clock time, or the overhead of curvature probes and SVD transfers. The experiments also do not include direct comparisons with the concurrent LAARA, GRIT, and CG-LoRA methods, so our distinctions from them are methodological rather than empirical. Finally, AdaLoRA's SVD parameterization stores more values than plain LoRA at the same nominal rank, so rank-matched comparisons are not parameter-matched; we report both accuracy and counts rather than conflating them.

8 Conclusion

CGRS treats LoRA rank as a capacity variable controlled by a task-gradient curvature signal. Across three ViT transfer benchmarks, its global scheduler improves over equal-final-rank LoRA on CIFAR-100 and SVHN and exceeds the tested AdaLoRA accuracies, while failing to do so on the smaller Flowers-102 benchmark. Layer-wise scheduling can save substantial final adapter storage—most notably on CIFAR-100—but one-way growth and permissive thresholds can collapse to uniform maximum rank. The next decisive experiments are multi-seed validation of the headline runs, shrinkable layer-wise allocation, threshold normalization or hysteresis, and direct measurement of training cost.

REPRODUCIBILITY STATEMENT

The anonymized implementation and experiment configurations are available in our anonymous code repository. Additional training details and complete experimental results are provided in the appendices.

AI USE STATEMENT

Generative AI tools were used to assist with literature discovery; drafting, editing, and organizing the manuscript; checking LaTeX formatting and references; refining figure captions; debugging and revising experimental code; and implementing the AdaLoRA comparison baseline using the PEFT library. The authors executed the experiments, generated the reported data and figures, and reviewed and verified all numerical results, citations, and claims against the underlying code, training logs, result files, and primary sources. The authors take full responsibility for the final content of this work, including all artifacts produced with generative-AI assistance.

REFERENCES

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL https://jmlr.org/papers/v21/17-678.html.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 722–729, 2008. doi: 10.1109/ICVGIP.2008.47.

Pritish Saha, Chandrav Rajbangshi, Rudra Goyal, Mohit Goyal, Anurag Deo, Biswajit Roy, Ningthoujam Dhanachandra Singh, Raxit Goswami, and Amitava Das. GRIT: Geometry-aware PEFT with K-FAC preconditioning, fisher-guided reprojection, and dynamic rank adaptation. *arXiv preprint arXiv:2601.00231*, 2026. URL https://arxiv.org/abs/2601.00231.

Ashutosh Tripathi, Surya Deep Singh, Pranab Sahoo, and Sriparna Saha. LAARA: Layer-aware adaptive rank allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2607.19391*, 2026. URL https://arxiv.org/abs/2607.19391.

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. URL https://aclanthology.org/2023.eacl-main.239/.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. IncreLoRA: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*, 2023a. URL https://arxiv.org/abs/2308.12043.

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. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=lq62uWRJjiY.

Frédéric Zheng and Alexandre Proutière. Curvature-guided LoRA: Steering in the pretrained NTK subspace. *arXiv preprint arXiv:2603.29824*, 2026. URL https://arxiv.org/abs/2603.29824.

A COMPLETE FIXED-RANK RESULTS

Table 3: Complete fixed-rank LoRA sweep. Parameters are adapter parameters and are identical across datasets at a given rank. Each accuracy is a single seed-42 run.

Rank	Params (M)	CIFAR-100	SVHN	Flowers-102
3	0.111	85.07	96.43	95.56
5	0.184	87.13	96.90	96.80
6	0.221	87.50	96.80	96.57
10	0.369	89.36	96.57	98.00
12	0.442	89.15	96.75	98.41
16	0.590	89.74	96.58	98.15
30	1.106	90.03	96.48	98.03
52	1.917	90.54	96.98	98.24
64	2.359	90.46	96.83	98.73
80	2.949	90.54	96.75	98.70
100	3.686	90.40	96.63	98.86
128	4.719	91.07	96.98	98.89
150	5.530	90.70	96.88	98.75
200	7.373	90.62	96.80	98.54

Table 4: Conventional baselines. Unlike adapter counts elsewhere, parameters here are all trainable parameters in the corresponding baseline.

Dataset	Full fine-tune (params; acc.)	Frozen backbone (params; acc.)
CIFAR-100	85,875,556; 92.55	76,900; 86.34
SVHN	85,806,346; 97.77	7,690; 55.06
Flowers-102	85,877,094; 99.01	78,438; 99.20

B All Global and AdaLoRA Runs

Table 5: All global CGRS runs. Parameters are final adapter parameters; number of rank transitions was not recorded in the supplied report for SVHN or Flowers-102.

Dataset	Threshold	Final r	Changes	Params (M)	Acc.
CIFAR-100	25th	64	1	2.359	90.72
	50th	64	6	2.359	90.62
	75th	64	8	2.359	**91.16**
SVHN	25th	52	–	1.917	97.03
	50th	64	–	2.359	97.21
	75th	52	–	1.917	**97.45**
Flowers-102	25th	64	–	2.359	**98.52**
	50th	64	–	2.359	98.28
	75th	30	–	1.106	97.01

Table 6: AdaLoRA comparison. Parameter counts reflect the stored over-parameterized adapter tensors associated with initialization at $1.5\times$ target rank.

Dataset	Target rank	Params (M)	Accuracy
CIFAR-100	16	0.885	89.25
	32	1.771	89.71
	64	3.541	90.06
SVHN	16	0.885	95.91
	32	1.771	95.74
	64	3.541	95.86
Flowers-102	16	0.885	98.55
	32	1.771	98.78
	64	3.541	98.60

C Training and Reproducibility Details

Table 7: Shared optimization settings.

Setting	Value
Backbone	`google/vit-base-patch16-224`
LoRA targets	query and value projections
α / dropout	16 / 0.1
Optimizer / weight decay	AdamW / 0.01
Learning-rate schedule	cosine annealing
LR (learning rate): full / head / LoRA	5×10^{-5} / 10^{-3} / 5×10^{-4}
Batch size / clipping	16 / global norm 1.0
Seed	42
Global initial rank	16
Layer-wise initial rank / bounds	30 / [3, 64]
Curvature probes	16 batches (profiling/layer-wise); 8–16 (global)

Phase 1 caches every fixed-rank result and checkpoint. Phase 2 caches global and adapter-wise curvature profiles. Global CGRS caches completed runs but cannot resume within a run. Layer-wise runs checkpoint after every epoch, including model state, ranks, cooldown counters, transition history, step, and learning rate. Results were consolidated per dataset for analysis.