Global Factorized Adaptation: Parameter-Efficient Fine-Tuning via Cross-Module Low-Rank Sharing

Anonymous Authors[1]

Abstract

We introduce *Global Factorized Adaptation* (GFA), a parameter-efficient fine-tuning primitive that replaces LoRA's N per-module low-rank factorizations with a single global factorization over the concatenated *weight update*. The substitution targets a previously underexplored operating point: LoRA-like per-step compute, competitive accuracy against a strong per-module baseline, and adapter budgets below the full-precision local rank-1 floor. Across 17 (model, task) cells covering 11 unique tasks on RoBERTa-large, Qwen-2.5-3B, and LLaMA-3.2-3B, GFA is competitive with DoRA at a matched number of parameters. At the same time, GFA matches LoRA's median tokens-per-second and VRAM, whereas DoRA pays a 24–74% wall-clock penalty. Under an equal-module approximation, GFA uses $O(r\sqrt{d})$ trainable parameters rather than LoRA's $O(r\sqrt{dN})$, yielding a $\sqrt{N}$ parameter-count advantage. This enables a sub-LoRA-$r = 1$ budget anchor, that uses roughly $0.3\times$ the parameters of LoRA $r = 1$ while retaining at least 97% of rank-16 accuracy on every evaluated decoder task.

1. Introduction

Parameter-efficient fine-tuning (PEFT) of foundation models has largely converged on the per-module low-rank update (Hu et al., 2022; Houlsby et al., 2019; Ding et al., 2023). For each target matrix $W_i \in \mathbb{R}^{p_i \times q_i}$, LoRA learns

$$\Delta W_i = (\alpha/r) B_i A_i, \qquad B_i \in \mathbb{R}^{p_i \times r}, \quad A_i \in \mathbb{R}^{r \times q_i}.$$

This design is fast and effective, but it fixes the unit of compression to be the module. If adaptation updates contain cross-layer or cross-module structure, then independent local factorizations may spend parameters redundantly while still imposing a hard rank cap on each module.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

GFA asks whether the low-rank constraint must be local. Instead of learning N independent factor pairs, we concatenate all target update coordinates, reshape them into one near-square matrix, and learn a single global factorization BA, and scatter the resulting update back to the original module shapes. The frozen pretrained weights are never factorized; only their additive perturbation is parameterized globally.

Relation to shared PEFT. Several recent methods reduce the parameter count of per-module PEFT by sharing or structuring components: VeRA shares frozen random matrices with learned scales (Kopiczko et al., 2024), Tied-LoRA ties factors across layers (Renduchintala et al., 2024), VB-LoRA uses a vector bank (Li et al., 2024), Uni-LoRA reconstructs LoRA parameters from shared latent vectors (Li et al., 2025), and RandLoRA uses frozen random bases to produce full-rank updates (Albert et al., 2025). GFA differs from all of these in what it factorizes: the trainable object is the concatenated weight-update vector itself, not a shared component inside the per-module LoRA parameterization. This is what enables the sub-LoRA-$r = 1$ budget regime, which is unreachable in the per-module template at full precision.

The $\sqrt{N}$ parameter-count argument. Let $d_i = p_i q_i$ and $d = \sum_i d_i$. Equal-rank LoRA uses

$$P_{LoRA} = r \sum_i (p_i + q_i).$$

For N equal-size square modules with $d_i = d/N$, this becomes

$$P_{LoRA} = 2r\sqrt{dN}.$$

GFA reshapes the concatenated update into $M \in \mathbb{R}^{m \times k}$ with $mk \geq d$ and $m \approx k \approx \sqrt{d}$, giving

$$P_{GFA} = r(m + k) \approx 2r\sqrt{d}.$$

Thus, under the equal-module approximation,

$$\frac{P_{LoRA}}{P_{GFA}} \approx \sqrt{N}. \tag{1}$$

At the $q + v$ attention scope used in our experiments, this gives a $7.5\times$–$9.7\times$ trainable-parameter saving across the three model families ($N = 48$ for RoBERTa-large, $N = 72$ for Qwen-2.5-3B, $N = 56$ for LLaMA-3.2-3B).

Contributions.

1. We propose GFA, a global low-rank PEFT primitive that moves the rank constraint from individual modules to the concatenated adaptation vector.
2. We identify a sub-LoRA-$r = 1$ operating point, denoted $r = 1/2$, that is unreachable by full-precision per-module adapters without quantization or weight tying.
3. We provide an approximation lens showing that GFA and LoRA are generally incomparable: global factorization is favored when the reshaped global update has the smaller budgeted spectral tail, while local LoRA is favored when updates are locally low-rank but globally incoherent.
4. We report a rank-anchor sweep over 17 (model, task) cells and an efficiency verification over 135 paired cells, showing LoRA-like compute and DoRA-comparable accuracy under a conservative tie-band convention, with strict wins on the most capacity-sensitive cells.

2. Global Factorized Adaptation

Consider a frozen model with target matrices $W_1 \in \mathbb{R}^{p_1 \times q_1}, \ldots, W_N \in \mathbb{R}^{p_N \times q_N}$ and total target dimension $d = \sum_i p_i q_i$. The frozen weights W_i are *not* factorized; only their additive perturbations ΔW_i are. GFA parameterizes the trainable perturbation in four steps.

Step 1: concatenate. Define the concatenated update coordinate vector

$$\delta = [\mathrm{vec}(\Delta W_1); \ldots; \mathrm{vec}(\Delta W_N)] \in \mathbb{R}^d.$$

Step 2: reshape. Choose $k = \lfloor \sqrt{d} \rfloor$, $m = \lceil d/k \rceil$, and view δ (with at most $k-1$ padding coordinates) as a matrix $M \in \mathbb{R}^{m \times k}$. Padding entries are discarded after scattering and receive no gradient signal.

Step 3: factorize. Learn factors $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times k}$, with $M = BA$. We initialize $B = 0$ and sample $A \sim \mathcal{N}(0, \sigma^2)$, so the initial update is 0 (mirroring LoRA).

Step 4: scatter. Let $c_i = \sum_{j \le i} p_j q_j$. The effective weights during training are

$$W_i^{\mathrm{eff}} = W_i + s \cdot \mathrm{reshape}\big(\mathrm{vec}(BA)_{c_{i-1}:c_i},\, p_i,\, q_i\big)\,,$$

with fixed scale $s = 1$. Algorithm 1 computes only the rows of BA intersecting each module slice.

The trainable count is $r(m+k) \approx 2r\sqrt{d}$, and the adapter FLOPs are $O(rd)$, matching equal-rank LoRA in order. Like LoRA, GFA merges into the base weights after training, so inference has no adapter overhead.

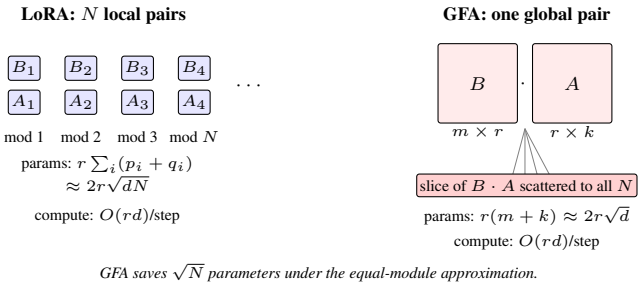

Figure 1. **Local vs. global.** LoRA gives each module its own pair. GFA uses a global pair and scatters the slices back to the modules.

Algorithm 1 GFA forward pass. The implementation computes only the slices. The $m \times k$ product is not materialized.

Input: frozen $\{W_i\}_{i=1}^N$, trainable $B \in \mathbb{R}^{m \times r}$, $A \in \mathbb{R}^{r \times k}$, scale s, boundaries $\{(c_i, p_i, q_i)\}$
for $i = 1, \ldots, N$ **do**
  $\delta_i \leftarrow \textsc{ExtractRows}(B, A, c_{i-1}, c_i, k)$
  $\mathrm{out}_i \leftarrow (W_i + s \cdot \mathrm{reshape}(\delta_i, p_i, q_i))x_i$
end for

Sub-LoRA-rank-1 budget anchor. Per-module low-rank adapters have a positive full-precision rank floor at $r{=}1$: LoRA $r{=}1$ has trainable count $\sum_i (p_i + q_i)$, which cannot be reduced without quantization or weight tying. GFA can operate below this floor by choosing an integer-valued global factorization whose trainable count lies below the LoRA $r{=}1$ budget. We denote the corresponding ultra-low-budget anchor by $r{=}1/2$ as a budget label only: it is shorthand for a configuration whose trainable count is roughly $0.3\times$ that of LoRA $r{=}1$ on the decoder settings, not a literal half-rank matrix factorization. All factor dimensions remain integer-valued; the label simply places this point on the same rank-normalized horizontal axis as LoRA.

Local rank is flexible, but not universally larger. LoRA enforces $\mathrm{rank}(\Delta W_i) \le r$ for every module. For example, a rank-16 LoRA update on a 1024×1024 attention matrix lies on a manifold of dimension $16(1024 + 1024 - 16) = 32{,}512$, far below the $1{,}048{,}576$ entries of an unconstrained full-rank update. GFA instead constrains only the rank of the global reshaped update. After scattering, slices of a low-rank global matrix can reshape into local module updates whose rank exceeds the nominal global rank: even a single row of BA can contain $p_i q_i$ entries that reshape into a full-rank $p_i \times q_i$ local matrix. Thus global rank 1 does not imply local rank 1 after scattering. GFA removes the explicit local rank cap while retaining a global coupling constraint; the actual realized local rank depends on the learned factors and the scatter geometry. We therefore frame GFA as a different inductive bias rather than a strict superset of LoRA.

3. Approximation Lens

The right theoretical comparison is not whether one hypothesis class contains the other. In general, neither does. The relevant question is which class gives a smaller approximation error under a fixed trainable budget.

Let $\Delta^\star = \{\Delta W_i^\star\}_{i=1}^N$ be a target update, and define

$$M^\star = \text{reshape}\left([\text{vec}(\Delta W_1^\star); \ldots; \text{vec}(\Delta W_N^\star)]\right) \in \mathbb{R}^{m\times k}.$$

Definition 3.1 (Budgeted errors)**.** For budget P, define

$$\mathcal{E}_{GFA}(P) = \min_{r:\, r(m+k)\le P} \min_{\text{rank}(M)\le r} \|M^\star - M\|_F^2,$$

and

$$\mathcal{E}_{LoRA}(P) = \min_{r_i:\, \sum_i r_i(p_i+q_i)\le P} \sum_i \min_{\text{rank}(X_i)\le r_i} \|\Delta W_i^\star - X_i\|_F^2.$$

Proposition 3.2 (Oracle approximation criterion)**.** *Let $\sigma_j(M^\star)$ denote the singular values of the global reshaped target, and $\sigma_{i,j}$ the singular values of $\Delta W_i^\star$. Then*

$$\mathcal{E}_{GFA}(P) = \min_{r:\, r(m+k)\le P} \sum_{j>r} \sigma_j(M^\star)^2,$$

whereas

$$\mathcal{E}_{LoRA}(P) = \min_{\{r_i\}:\, \sum_i r_i(p_i+q_i)\le P} \sum_i \sum_{j>r_i} \sigma_{i,j}^2.$$

Thus GFA is favored when the globally reshaped update has a smaller budgeted spectral tail than the best budgeted collection of local low-rank approximations.

Proof. Vectorization and reshaping preserve the Euclidean norm, so GFA's oracle is the best rank-r approximation of $M^\star$, minimized over budget-feasible ranks. By Eckart–Young–Mirsky, the rank-r error is $\sum_{j>r} \sigma_j(M^\star)^2$. For LoRA, each module is approximated independently at rank r_i, and applying the same theorem module-wise gives the second expression. □

Falsifiable prediction. If a high-quality proxy update, such as a full fine-tuning delta or high-rank adapter delta, satisfies

$$\mathcal{E}_{GFA}(P) < \mathcal{E}_{LoRA}(P),$$

then global factorization has a representational explanation at budget P. If the inequality reverses but GFA still wins, the advantage must come from optimization, regularization, or mismatch between the proxy update and the learned solution. This makes the mechanism testable rather than assumed.

Incomparability. There are updates that are rank-one in every local module but high-rank after global reshaping; those favor LoRA. Conversely, a rank-one global matrix can contain a row whose entries reshape into a high-rank local module update; those favor GFA. We therefore frame GFA as a different inductive bias, not as a universal improvement over local adapters.

4. Decoder Families: Qwen and LLaMA

Figure 2 compares GFA, LoRA, and DoRA on Qwen-2.5-3B-Instruct and LLaMA-3.2-3B-Instruct across six tasks (`arc_easy`, `arc_challenge`, `boolq`, `obqa`, `piqa`, `winogrande`). The main observation is that GFA's curve is competitive with DoRA's at every matched rank anchor, with strict outside-band wins on capacity-sensitive cells, while retaining LoRA-like per-step compute (Sec. 6). Many at-rank gaps fall inside the ± 1.4pp tie band defined in Sec. 7; these are reported as ties rather than wins.

Matched-rank behavior. On Qwen, GFA reaches 87.8 on `piqa` at r=16, compared with DoRA's 87.3 (+0.5pp, inside the tie band). On `arc_challenge`, DoRA leads at the lowest rank but GFA closes the gap and reaches 84.8 at r=16, compared with DoRA's 83.1 (+1.7pp, a strict outside-band win). On LLaMA, GFA reaches 85.8 on `obqa` at r=16 versus DoRA's 84.4 (+1.4pp, exactly at the tie-band boundary; we report this as a tie rather than as a strict win to be conservative). Several cells (Qwen `obqa`/`arc_easy` at $r\ge 4$, LLaMA `piqa`/`boolq` at $r\ge 8$, `winogrande` on LLaMA across all ranks) are within the seed band; we treat those as ties. Counted strictly (excluding ties), GFA's at-rank wins concentrate at $r \in \{1, 2, 16\}$; the middle ranks $r \in \{4, 8\}$ show higher tie rates as both methods saturate. The unambiguous outside-band wins at r=16 on the decoder panels are Qwen `arc_challenge` and a closing gap on Qwen `winogrande`; the encoder section (Sec. 5) adds RoBERTa `cola` and `rte`.

Sub-LoRA-$r = 1$ retention. The leftmost GFA point on each panel is the $r = 1/2$ budget anchor. LoRA's minimum positive rank is 1, about 230K parameters on Qwen $q+v$ and 287K on LLaMA $q+v$. The GFA anchor lies at roughly 70K parameters, roughly $0.3\times$ of LoRA $r = 1$. Retention from rank-16 to $r = 1/2$ is high across all twelve panels, with every task retaining at least 97% of its rank-16 accuracy. The three largest drops are LLaMA `winogrande` (-2.5pp), LLaMA `piqa` (-1.6pp), and Qwen `arc_challenge` (-1.5pp); the remaining decoder tasks drop by at most 1.2pp. For full visibility, Qwen drops are `arc_easy` -0.3pp, `boolq` -0.6pp, `winogrande` -1.0pp, `piqa` -1.2pp, `arc_challenge` -1.5pp, and `obqa` -1.0pp; LLaMA drops are `arc_easy` -0.5pp, `boolq` -0.6pp, `arc_challenge` -0.7pp, `obqa` -0.6pp, `piqa` -1.6pp, and `winogrande` -2.5pp.

5. Encoder Family: RoBERTa-large

Figure 4 reports five GLUE tasks: `cola`, `mnli`, `mrpc`, `rte`, and `sst2`. The encoder has $N = 48$ target matrices at $q + v$ scope. Following standard GLUE practice, all methods share the classifier head; we therefore plot adapter-only counts.

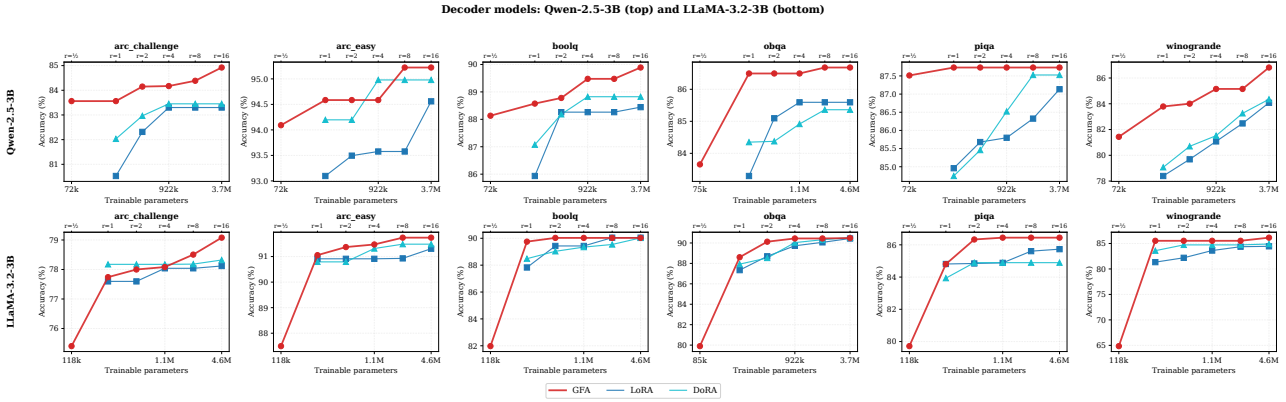

Figure 2. **Two decoder families, six tasks each.** Top row: Qwen-2.5 ($N = 72$, $d \approx 226$M). Bottom row: LLaMA-3.2 ($N = 56$, $d \approx 201$M). Methods are placed at the parameter counts induced by LoRA ranks $\{1, 2, 4, 8, 16\}$. GFA additionally carries the $r = 1/2$.

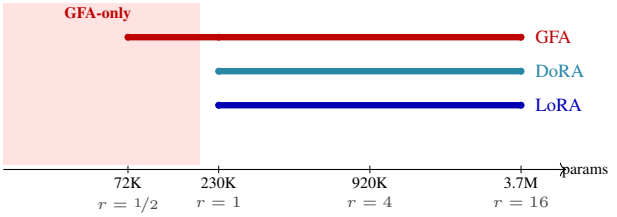

Figure 3. **Operating regime.** Per-module methods are bounded below by their integer-rank floor at $r = 1$. GFA enters the shaded sub-LoRA-$r = 1$ region by adjusting the global factorization size.

CoLA: a capacity-sensitive task. `cola`'s Matthews correlation rises from 52.9 at LoRA $r = 16$ to 67.6 at GFA $r = 16$, a +14.7pp gap; DoRA reaches 55.1, leaving GFA +12.5pp ahead. One possible interpretation is that CoLA benefits from update capacity beyond the local rank-16 bottleneck. We do not claim this is proven: the local-rank interpretation is a capacity heuristic, and the direct diagnostic would be the singular-value spectrum of full fine-tuning or high-rank adapter deltas under the global reshape, which is exactly the test prescribed by Proposition 3.2.

Saturated GLUE tasks and SST-2. On `mnli`, `mrpc`, and `sst2`, high-rank differences are small. In particular, `sst2` has a numerical −0.3pp difference against DoRA at $r = 16$, which is well inside the tie band and is not counted as a meaningful loss. `rte` shows a moderate +3.9pp advantage for GFA at $r = 16$ (a strict win), but its small validation set makes variance larger than on the decoder tasks.

Sub-LoRA-$r = 1$ regime on encoders. The $r = 1/2$ anchor on RoBERTa is about 11K trainable adapter parameters, excluding the shared $\sim$ 1M classifier head. If the head is included, the total trainable count is dominated by the head, and the apparent savings shrink. On `sst2`, `mnli`, and `mrpc`, GFA-1/2 stays within roughly 1pp of its rank-16 accuracy. On `rte`, the drop is about 4pp. On `cola`, the low-budget adapter is near-degenerate, consistent with CoLA being the most capacity-sensitive task in this sweep.

Table 1. **Same-parameter-count efficiency.** At a matched trainable-parameter LoRA $r = 4$, GFA matches LoRA on VRAM and throughput while DoRA pays a penalty.

Method	Trainable	Peak VRAM	Throughput	Adapter FLOPs
LoRA	1.00×	1.000×	1.000×	1.00×
DoRA	1.00×	1.06×	0.85×	≈ 2×
GFA	0.62×	0.999×	1.000×	1.00×
GFA-R1	0.30×	0.97×	0.98×	1.00×

6. Compute and Memory

Headline. GFA's per-step footprint is indistinguishable from LoRA's in the median: TPS ratio 1.000, VRAM ratio 0.999 over 135 paired cells. The worst observed slowdown is a single 0.91× TPS regression.

Directional breakdown. Across 135 paired cells, 0/135 have GFA VRAM exceeding LoRA by at least 5%. In contrast, 15/135 have at least 5% lower VRAM than LoRA, concentrated on encoder/vision backbones. On throughput, 7/135 cells have GFA TPS at least 5% above LoRA, while 6/135 cells are at least 5% below. On Qwen and LLaMA decoders, GFA tracks LoRA within roughly ±1% in the median. DoRA's wall-clock penalty over the same measurement suite is 24–74% (median $\sim$ 30%), attributable to magnitude/direction rescaling.

7. Discussion and Limitations

Tie-band convention. Multi-seed studies on representative Qwen tasks and ViT CIFAR-100 give a pooled standard deviation of approximately 0.7pp, implying a rough ±1.4pp 95% interval on large-dev-set accuracy-style cells. We use this as a conservative tie band: gaps inside the band are parities, not strict wins. We do not apply the band mechanically to small-dev-set GLUE tasks (RTE, MRPC) or to Matthews correlation on CoLA, where seed variance is likely larger.

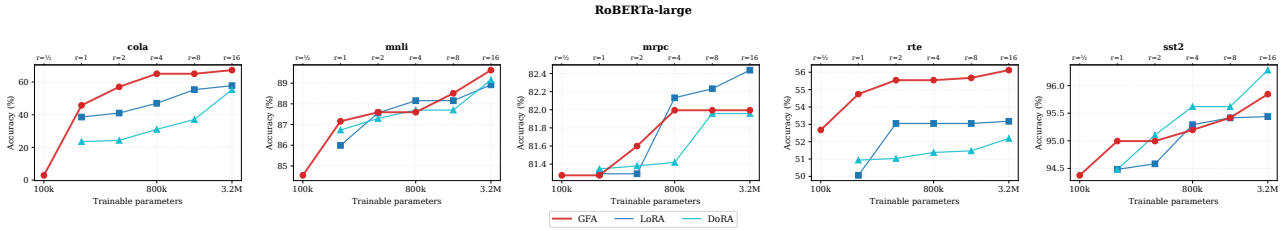

Figure 4. **RoBERTa/GLUE.** Same axes and methods as Figure 2. The x-axis is adapter-only and excludes the head shared by all methods.

Measured claims vs. conjectured mechanisms. Three claims are directly measured or analytical: the $\sqrt{N}$ parameter saving under the equal-module approximation, per-step compute parity with LoRA, and competitive rank-anchor accuracy against DoRA under the tie convention. Two mechanism claims remain conjectural: the capacity explanation for CoLA and the spectral interpretation of permutation insensitivity. The approximation lens in Sec. 3 gives a direct way to test these claims using proxy update spectra.

Baseline tuning. LoRA is the canonical compute baseline, and DoRA is a strong accuracy-oriented baseline within the same per-module template. We use matched training settings rather than method-specific learning-rate and dropout sweeps, did not tune each baseline more exhaustively.

Scope. The headline sweep covers models up to 3B parameters and focuses on $q + v$ adapter scope. We do not include a full 7B+ instruction-tuning sweep or a broad vision suite. ViT CIFAR-100 is used for permutation and efficiency diagnostics but is not included in the main rank-anchor figures.

Reproducibility. We will release scripts for parameter counting, rank-anchor construction, efficiency measurement, and all training configurations. The method itself requires no custom kernels; the core implementation is the chunked extraction of rows from BA, as shown in Algorithm 1.

8. Conclusion

GFA replaces N per-module low-rank factorizations with one global factorization over the concatenated adaptation update. Under the equal-module approximation, this changes the trainable count from $O(r\sqrt{dN})$ for equal-rank LoRA to $O(r\sqrt{d})$, opening an ultra-low-budget regime below the local rank-1 floor of per-module full-precision adapters. In our rank-anchor sweep over 17 (model, task) cells covering 11 unique tasks, GFA matches LoRA's per-step compute and remains competitive with DoRA under a conservative tie convention, with strict outside-band wins on capacity-sensitive cells. At the sub-LoRA-$r = 1$ budget anchor, GFA retains at least 97% of rank-16 decoder accuracy on every evaluated decoder task. The main open questions are multi-seed confirmation, the direct spectral diagnostic prescribed by Proposition 3.2, and scaling to 7B+ instruction tuning.

References

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL*, 2021.

Albert, P., Zhao, F. Z., and Bansal, M. RandLoRA: Full-rank parameter-efficient fine-tuning of large models. In *ICLR*, 2025.

Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5:220–235, 2023.

Gao, Z., Wang, Q., Chen, A., Liu, Z., Wu, B., Chen, L., and Li, J. FourierFT: Parameter-efficient fine-tuning with discrete Fourier transform. In *ICML*, 2024.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Kopiczko, D. J., Blankevoort, T., and Asano, Y. M. VeRA: Vector-based random matrix adaptation. In *ICLR*, 2024.

Li, Y., Han, S., and Ji, S. VB-LoRA: Extreme parameter efficient fine-tuning with vector banks. In *NeurIPS*, 2024.

Li, K., Han, S., Su, Q., Li, W., Cai, Z., and Ji, S. Uni-LoRA: One vector is all you need. In *NeurIPS*, 2025.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. DoRA: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

Renduchintala, A., Konuk, T., and Kuchaiev, O. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In *NAACL*, 2024.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.