SOLAR: COMMUNICATION-EFFICIENT MODEL ADAPTATION VIA SUBSPACE-ORIENTED REPARAMETRIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning (PEFT) methods, such as LoRA, enable scalable adaptation of foundation models by injecting low-rank adapters. However, their communication and storage costs remain a major bottleneck in resource-constrained settings. We propose **SOLAR** (Subspace-Oriented Latent Adapter Reparameterization), a post-training compression framework that substantially reduces the communication cost (i.e., the number of parameters to transmit or store) of PEFT adapters. SOLAR expresses each PEFT update as a linear combination of basis vectors formed from the foundation model's singular vectors with controlled random perturbations. By exploiting the subspace similarity (the alignment of principal directions) between the foundation model and task-specific fine-tuned updates, SOLAR decouples the adapter size from PEFT structure and ensures compact yet expressive representations. It is model-agnostic and compatible with existing PEFT methods, including LoRA and other adapter modules. We theoretically establish a bound on the reconstruction error. Experiments on language and vision tasks using LLaMA, GPT, and ViT models demonstrate that SOLAR preserves task performance while significantly reducing model representation sizes, offering an effective and communication-efficient solution for deployment in distributed systems and edge devices.

1 INTRODUCTION

Foundation models—large-scale pretrained transformer architectures—have catalyzed substantial progress across natural language processing, computer vision, and a range of other domains. However, adapting these models to downstream tasks remains resource-intensive. Full fine-tuning, which updates all model parameters, demands considerable computational, memory, and storage resources Houlsby et al. (2019). Parameter-Efficient Fine-Tuning (PEFT) techniques address this challenge by freezing the backbone and updating only a small set of task-specific parameters. For example, adapter modules insert compact trainable layers into each network block Houlsby et al. (2019); prefix-tuning optimizes a continuous prompt of only $\sim$0.1% of the model's parameters Li & Liang (2021); and Low-Rank Adaptation (LoRA) injects low-rank update matrices into each layer Hu et al. (2021). These methods achieve performance comparable to fully fine-tuned models while updating less than 1% of the model's parameters.

Despite these parameter savings, the cumulative communication and storage costs of PEFT modules remain a critical bottleneck in many real-world scenarios, particularly as foundation models continue to scale Wolf et al. (2020). In distributed scenarios (e.g., federated learning), these adapters must be communicated and stored across multiple devices or nodes, leading to significant overhead Wolf et al. (2020). Communication and storage overhead increase with the number of PEFT modules, as many fine-tuned adapters are saved and frequently transmitted or synchronized, thus turning millions of adapter parameters into a major bottleneck, particularly in bandwidth-limited or memory-constrained environments such as edge devices or federated learning systems Gao & Zhang (2024); Wang et al. (2025). The resulting communication and storage costs (i.e., the number of adapter parameters that must be transmitted and stored) can lead to slower training, increased energy consumption, and reduced scalability, highlighting the need for more efficient adapter compression techniques.

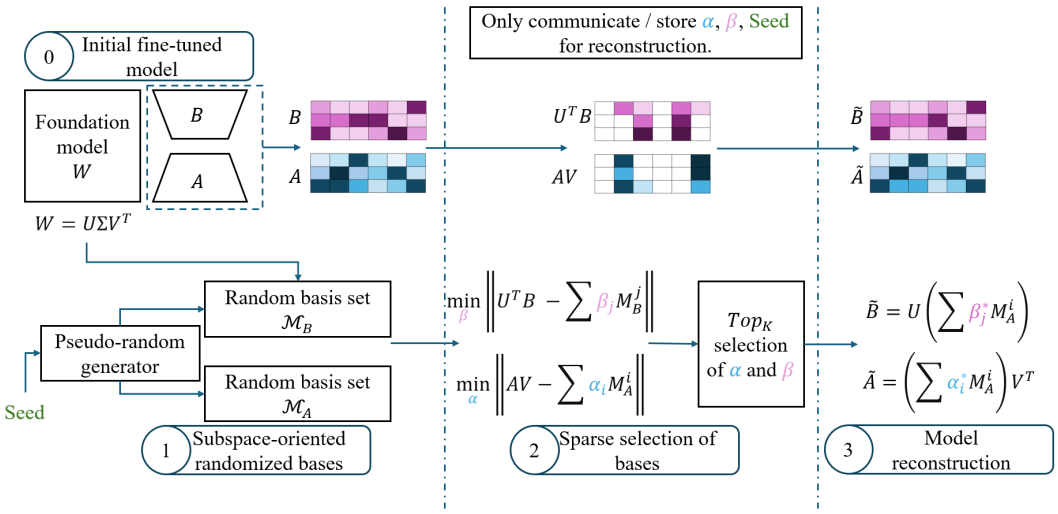

Figure 1: Overview of SOLAR. Given fine-tuned adapters (A, B), SOLAR projects them onto structured subspaces derived from the pretrained model's SVD. A pseudo-random generator, seeded with a known value, constructs partially randomized basis matrices. Top_k coefficients α and β are selected under a communication/storage budget to reconstruct the compressed update matrices $\tilde{A}$ and $\tilde{B}$. Only the coefficients α, β, and the seed need to be communicated or stored.

To address this, several methods decouple tunable parameters from adapter rank and model dimensions: NOLA Koohpayegani et al. (2024) expresses LoRA's matrices as linear combinations of random basis matrices, training only the coefficients; VeRA Kopiczko et al. (2023) uses shared frozen random vectors with small learned scaling vectors; and SVFT Lingam et al. (2024) constructs a basis from singular vectors of pretrained weights and learns a sparse combination during fine-tuning. However, random bases not aligned with the model or task may reduce representational efficiency, and methods such as Kopiczko et al. (2023); Lingam et al. (2024); Koohpayegani et al. (2024) are not post-hoc, as they modify the training process and cannot compress adapters already trained—creating a need for a flexible, training-free compression utility.

In this paper, we propose SOLAR (**S**ubspace-**O**riented **L**atent **A**dapter **R**eparameterization), a novel post-training compression method for PEFT adapters. SOLAR exploits the empirical structure of adapter updates by reparameterizing them as linear combinations of structured, randomized basis matrices. It is model-agnostic and applicable post-training without modifying the fine-tuning process. The main contributions of this work are as follows:

- We leverage the observed subspace similarity between the foundation model's weights (W) and the task-specific update (ΔW) to create a more compact and efficient adapter representation. By expressing ΔW as a sparse combination of basis vectors, our method effectively decouples the adapter's final size from the model's architecture.
- We develop a three-step framework for post-hoc adapter compression that involves: 1) constructing a basis pool of size N by perturbing the foundation model's singular vectors with random noise, 2) performing a sparse selection of the most significant basis vectors to meet a budget k, and 3) reconstructing the adapter using only the selected coefficients and a single random seed.
- We provide a formal theoretical analysis that bounds the reconstruction error. Our proof decomposes the total error into the original training error and a controllable compression error, which can be minimized by tuning SOLAR's hyperparameters (N and k).
- We demonstrate through extensive experiments that SOLAR reduces adapter sizes by up to 98% while preserving the performance of the original LoRA adapters. Our results show competitive accuracy across a wide range of vision and language tasks using ViT, GPT-2, and LLaMA models.

2 Proposed Method: SOLAR

We propose a *post-training* compression strategy that serves as a modular add-on for compressing PEFT-based updates. It introduces no training overhead and is compatible with LoRA Hu et al. (2021),

QLoRA Dettmers et al. (2023), Compacter Karimi Mahabadi et al. (2021), and NOLA Koohpayegani et al. (2024), operating post-hoc by taking the final trained adapter matrices as input. SOLAR also applies to Orthogonal Finetuning (OFT) Qiu et al. (2023) and variants Liu et al. (2023), as its SVD-based subspace captures principal directions and compresses orthogonal matrices post-hoc. By exploiting the low-rank structure of updates, SOLAR significantly reduces communication and storage costs in distributed or resource-limited settings.

2.1 Problem Formulation

Transformer-based models parameterize attention and MLP layers using full-rank weight matrices $W \in \mathbb{R}^{m\times n}$. Recent PEFT methods, such as LoRA Hu et al. (2021), decompose the task-specific update ΔW as $\Delta W = BA$, where $A \in \mathbb{R}^{r\times n}, B \in \mathbb{R}^{m\times r}$, and $r \ll \min(m,n)$. This reduces the trainable parameters from mn to $r(m+n)$, yielding a compression ratio of $\frac{mn}{r(m+n)}$. While effective, LoRA's fixed-rank formulation limits its flexibility. Alternatives, such as NOLA Koohpayegani et al. (2024), leverage random projections to approximate ΔW, but often require large basis sets to sufficiently capture the relevant directions. To address this challenge and enhance compression further, we formulate the problem as minimizing the approximation loss between ΔW and its compressed counterpart $\Delta\tilde{W}$ subject to a strict communication (or storage) budget:

$$\min_{\Delta\tilde{W}} \|\Delta W - \Delta\tilde{W}\|_F^2, \quad \text{s.t. } \|\Delta\tilde{W}\|_0 \le k, \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and $\|\cdot\|_0$ represents the number of non-zero elements (i.e., $\|X\|_0 \triangleq \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbb{I}\{X_{ij} \neq 0\}$). The parameter k specifies the total budget.

Building on the LoRA formulation, we approximate the individual factors A and B, aiming to find compressed counterparts $\tilde{A}$, $\tilde{B}$ such that:

$$\min_{\tilde{A},\tilde{B}} \|BA - \tilde{B}\tilde{A}\|_F^2, \quad \text{s.t. } \|\tilde{A}\|_0 \le k_A,\ \|\tilde{B}\|_0 \le k_B, \quad k_A + k_B = k, \tag{2}$$

where k_A and k_B represent budgets for $\tilde{A}$ and $\tilde{B}$, respectively. This problem is challenging: counting the number of nonzero elements is non-convex, sparse element selection is combinatorial, and excessive sparsity may degrade accuracy. Achieving high compression without task performance loss thus requires careful subspace design and adaptive optimization.

2.2 Method: Subspace-Oriented Randomized Basis, Sparse Selection, and Reconstruction

To solve (2), we propose SOLAR. A key insight motivating our approach is that ΔW predominantly resides in the subspace spanned by W, particularly in LoRA-based fine-tuning, where constraining the rank $r \ll \min(m,n)$ forces ΔW to concentrate its variation along specific directions of W Hu et al. (2021). This alignment (i.e., the overlap in the principal directions of W and ΔW) has been observed empirically and explained theoretically via neural tangent kernel (NTK) theory Jacot et al. (2018); Malladi et al. (2023); Seleznova et al. (2023). The left- and right-singular alignments are measured as $\|U_W^\top U_{\Delta W}\|_F^2$ and $\|V_W^\top V_{\Delta W}\|_F^2$, where U and V contain the left and right singular vectors from the SVD of each matrix Hu et al. (2021). Under this perspective, the model's response to updates is well-approximated by a first-order expansion: $f(\xi; W + \Delta W) \approx f(\xi; W) + \langle \nabla f(\xi; W), \Delta W\rangle$, where f is the model, ξ is input data, and $\nabla_W f(\xi; W)$ denotes the gradient of the foundation model's output. This implies that ΔW lies in a low-curvature (and hence low-dimensional) subspace defined by W's parameter space (see Section 3.4 for empirical evidence). Thus, projecting ΔW into the subspace of W enables an efficient and compact representation that can be sparsified with minimal information loss.

Building on these insights, we design a three-stage compression framework (Figure 1). First, we construct a randomized basis set aligned with the foundation model (Section 2.2.1). Next, we select a sparse set of bases to approximate the projected update (Section 2.2.2). We then reconstruct the update using a budget-aware combination of selected components (Section 2.2.3).

2.2.1 Step 1: Subspace-Oriented Randomized Basis Set

We construct a basis set from the foundation model's parameter space via SVD of the model weight, $W = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthonormal, and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal. This

decomposition enables a basis naturally aligned with the directions of task-specific updates ΔW. Unlike methods such as NOLA Koohpayegani et al. (2024) relying on unstructured random bases, our foundation-aligned directions allow a more compact representation of ΔW.

To enrich the expressive power of this subspace, we construct randomized basis matrices by perturbing slices of the singular vectors:

$$\mathcal{M}_A = \left\{ M_A^{(i)} = V[:, \mathcal{I}_i] + \epsilon_i \right\}_{i=1}^{N_A}, \quad \mathcal{M}_B = \left\{ M_B^{(j)} = U[:, \mathcal{J}_j] + \epsilon_j \right\}_{j=1}^{N_B}, \tag{3}$$

where $\mathcal{I}_i$ and $\mathcal{J}_j$ are randomly sampled index sets, N_A, N_B are the number of basis candidates for A and B, respectively, and ϵ_i, ϵ_j are Gaussian noise $\mathcal{N}(0, 1)$. These basis sets form a flexible pool of candidates for approximation.

2.2.2 Step 2: Sparse Selection of Bases

To enable more compact approximations, the LoRA update $\Delta W = BA$ is first projected into the subspace of W. Given the singular value decomposition $W = U\Sigma V^T$, this projection is defined as $\Delta W_{\text{Proj}} = U^T \Delta W V = (U^T B)(AV) = B_{\text{Proj}} A_{\text{Proj}}$, where $A_{\text{Proj}} = AV$ and $B_{\text{Proj}} = U^T B$ represent the update components expressed in the basis of W. This transformation retains all information when W is full-rank, and is particularly effective when ΔW is already aligned with the foundation subspace, a property commonly observed in LoRA-based fine-tuning. Under this projection, the update becomes $\Delta W = U \Delta W_{\text{Proj}} V^T$. This approach leverages the inherent alignment between W and ΔW, enabling more efficient approximations with fewer basis elements than methods such as NOLA, which rely on unstructured random projections. Specifically, we approximate the projected LoRA factors AV and $U^T B$ using sparse linear combinations of the basis matrices:

$$\min_{\alpha} \left\| AV - \sum_{i=1}^{N_A} \alpha_i M_A^{(i)} \right\|_F^2, \text{s.t. } \|\alpha\|_0 \le k_A, \quad \min_{\beta} \left\| U^T B - \sum_{j=1}^{N_B} \beta_j M_B^{(j)} \right\|_F^2, \text{s.t. } \|\beta\|_0 \le k_B. \tag{4}$$

A two-step strategy is employed to solve these NP-hard problems efficiently. The first step computes the unconstrained least squares solution to obtain coefficients α^* and β^*. The second step applies hard thresholding to retain only the top$_k$ entries by magnitude based on the budgets k_A and k_B.

2.2.3 Step 3: Budget-Aware Reconstruction

The approximated model update is then reconstructed using the selected top$_k$ bases, resulting in $\tilde{A}$ and $\tilde{B}$ for A and B, respectively:

$$A \approx \left(\sum_{i \in S_A} \alpha_i^* M_A^{(i)} \right) V^T, \qquad B \approx U \left(\sum_{j \in S_B} \beta_j^* M_B^{(j)} \right), \tag{5}$$

where S_A and S_B are the selected top$_k$ index sets. Because the update reconstruction is performed within the subspace defined by W, this step ensures strong alignment with task-relevant directions. The reconstruction balances accuracy and compression, with the sparsity budgets k_A and k_B controlling the number of active basis.

Adaptive Compression. SOLAR enables flexible allocation of sparsity budgets k_A and k_B, adapting to system constraints such as memory, storage, or bandwidth. This allows deployment on resource-constrained devices, with adapter size dynamically adjustable post-training. For instance, a server can send a compact adapter to low-memory clients and a richer version to more capable devices.

2.3 Theoretical Analysis of Reconstruction Error

We assume that (A1) the model is initialized with spectral initialization; (A2) the optimal update is low-rank; (A3) the change in the model's weights from fine-tuning is well-behaved according to the generation process in Zhang et al. (2025a); and (A4) the singular values of the projected update matrix exhibit Fast Spectrum Decay. These assumptions are well-established and frequently utilized in the literature for convergence analyses, as in previous works, such as Zhang et al. (2025a); Martinsson & Tropp (2020).

Theorem 1 [SOLAR Reconstruction Error Bound] Let ΔW^* be the optimal low-rank adapter, ΔW be the adapter learned via fine-tuning, and $\Delta\tilde{W}$ be the adapter reconstructed by SOLAR. Under assumptions (A1)–(A4), the expected total error is bounded by $\mathbb{E}\left[\|\Delta\tilde{W} - \Delta W^*\|_F\right] \leq C_1 + C_2$, where C_1 captures the fine-tuning error (depending on learning rate, training steps, and spectrum of ΔW^*; see Appendix A), and $C_2 = \sqrt{1 + \frac{r}{N-r-1}}\sqrt{\sum_{i=r+1}^{N}\sigma_i^2(\Delta W)} + \sqrt{\sum_{j=k+1}^{N}\tilde{\sigma}_j^2} + \sqrt{mn}\,\|\beta\|_2\|\alpha\|_2$, with σ_i denoting the i-th singular value of the projected update ΔW and $\tilde{\sigma}_j$ denoting the singular values of the projected SOLAR matrix.

The SOLAR reconstruction error consists of two components: a training error from fine-tuning (C_1) and a compression error introduced by SOLAR (C_2). The compression error can be reduced by enlarging the basis pool N to lower projection error and increasing the sparsity budget k to minimize sparsification error. Detailed derivations and proofs are provided in Appendix A.

3 Experiments

We evaluate SOLAR through extensive experiments in three domains: 1) image classification with ViT-B/L in few-shot and full-data settings (Section 3.1); 2) instruction tuning on LLaMA-3 models using Alpaca and MMLU (Section 3.2); and 3) language generation with GPT-2 on E2E NLG (Section 3.3). Across all settings, SOLAR matches LoRA and NOLA in accuracy while reducing adapter size by up to 98%, offering a lightweight representation for model adaptation.

3.1 SOLAR on Vision Transformers

We conduct few-shot image classification experiments using ViT-B and ViT-L Dosovitskiy et al. (2020) foundation models, initialized with either supervised or self-supervised He et al. (2022).

Experimental Setup. We compare SOLAR against LoRA Hu et al. (2021) and NOLA Koohpayegani et al. (2024). Experiments are conducted on ViT-Base (ViT-B) and ViT-Large (ViT-L) architectures. Supervised ViT models pretrained on ImageNet-21k Deng et al. (2009) are obtained from Google's official releases via the Hugging Face repository Wolf et al. (2020); Research (2025), and MAE models pretrained on ImageNet-1K are sourced from the Timm library Wightman (2025). All experiments run on a single NVIDIA RTX 4090 GPU using PyTorch Paszke (2019) and HuggingFace libraries. In SOLAR, the compressed representation consists of (i) a random seed to regenerate the basis vectors, (ii) an encoded list of selected basis indices, and (iii) their coefficients. Reported trainable parameters include both projection coefficients and overhead (i.e., seed and index encoding). The MLP classifier head is dataset-specific and excluded from the parameter count unless noted.

Evaluation Benchmarks. We fine-tune on standard image classification datasets: CIFAR-10 Krizhevsky et al. (2009), CIFAR-100 Krizhevsky et al. (2009), Food-101 Bossard et al. (2014), Tiny-ImageNet Le & Yang (2015), ImageNet-1K Deng et al. (2009), Oxford Pets Parkhi et al. (2012), SUN397 Xiao et al. (2010), and CUB-200-2011 Welinder et al. (2010).

Comparison Methods. We compare SOLAR with several baselines: Full Fine-Tuning (Full-FT), LoRA Hu et al. (2021), and NOLA Koohpayegani et al. (2024). In Full-FT, all backbone parameters are updated. For LoRA, we apply low-rank adapters to the attention Query projection matrices, with a rank of 4 for ViT-B and either 1 or 4 for ViT-L. For NOLA, following Koohpayegani et al. (2024), adapters are inserted into MLP layers using 1000 random basis vectors for each of the A and B matrices. All models are trained with cross-entropy loss. For full-data settings, we train 5 epochs with batch size 128; for few-shot settings (10 samples per class), 25 epochs with batch size 16, emphasizing low-data efficiency relevant to real-world and distributed scenarios. To account for variance from limited data, we sample four training splits per dataset and report mean top-1 accuracy on the test split (or validation for ImageNet-1k). Experiments are repeated with different random seeds, and learning rates are tuned per dataset and model. Additional details are in the appendix.

Results and Performance Analysis. We evaluate SOLAR on various vision benchmarks using foundation models, with results in Table 1. In the tables, configurations are denoted as $\text{SOLAR}_{\text{method}(N\to k)}$, indicating that SOLAR is applied to a NOLA or LoRA model trained with rank r, using N bases per matrix ($N = N_A = N_B$) and selecting the top-k bases by significance, where N and k are given in thousands. SOLAR consistently achieves competitive top-1 accuracy in few-shot (10 samples per

Table 1: Top-1 classification accuracy (%) of ViT-B and ViT-L on benchmark datasets under two settings: (1) few-shot (10 samples/class, 25 epochs) and (2) full-data (5 epochs). Results report mean ± std over 5 runs. SOLAR is applied with configuration $_{\text{method}(N\to k)}$, where N and k are in thousands.

Model	Method	# Param	CIFAR-10 10	CIFAR-10 Full	CIFAR-100 10	CIFAR-100 Full	Food-101 10	Food-101 Full	T-ImageNet 10	T-ImageNet Full
ViT-B	Full-FT	86M	$91.1_{\pm.8}$	$94.6_{\pm.5}$	$78.2_{\pm.7}$	$87.7_{\pm.3}$	$65.8_{\pm.9}$	$85.2_{\pm.4}$	$78.1_{\pm1.0}$	$85.4_{\pm.6}$
	LoRA (r=4)	74K	**92.3**$_{\pm.6}$	**98.3**$_{\pm.2}$	**81.8**$_{\pm.8}$	**90.3**$_{\pm.4}$	**72.4**$_{\pm.7}$	**87.6**$_{\pm.3}$	$77.9_{\pm.9}$	**88.8**$_{\pm.4}$
	NOLA	48K	$92.2_{\pm.6}$	$94.7_{\pm.5}$	$81.3_{\pm.8}$	$86.6_{\pm.4}$	$72.6_{\pm.5}$	$85.9_{\pm.2}$	**78.4**$_{\pm.7}$	$82.8_{\pm.5}$
	SOLAR$_{r=4(4\to1.6)}$	41K	**92.3**$_{\pm.7}$	**98.3**$_{\pm.4}$	$81.5_{\pm.7}$	$89.8_{\pm.2}$	$71.8_{\pm.6}$	$87.0_{\pm.5}$	$77.9_{\pm.8}$	$87.9_{\pm.4}$
	SOLAR$_{\text{NOLA}(4\to1.2)}$	**32K**	$92.1_{\pm.7}$	$94.5_{\pm.3}$	$81.1_{\pm.6}$	$85.4_{\pm.3}$	$72.5_{\pm.6}$	$85.4_{\pm.3}$	$78.3_{\pm.8}$	$82.3_{\pm.5}$
ViT-L	Full-FT	303M	$90.2_{\pm.9}$	$94.1_{\pm.6}$	$86.2_{\pm.7}$	$87.7_{\pm.5}$	$73.9_{\pm.8}$	$85.5_{\pm.4}$	$80.8_{\pm1.1}$	$89.2_{\pm.6}$
	LoRA (r=4)	197K	**97.1**$_{\pm.5}$	**98.7**$_{\pm.1}$	**88.1**$_{\pm.7}$	$92.4_{\pm.3}$	$81.8_{\pm.7}$	**89.8**$_{\pm.2}$	**84.4**$_{\pm.8}$	**91.8**$_{\pm.5}$
	LoRA (r=2)	98K	$96.6_{\pm.4}$	**98.7**$_{\pm.1}$	**88.0**$_{\pm.6}$	**92.9**$_{\pm.3}$	$82.1_{\pm.7}$	**90.0**$_{\pm.2}$	$83.8_{\pm.7}$	$90.4_{\pm.3}$
	NOLA	96K	$96.0_{\pm.8}$	$97.4_{\pm.6}$	$87.8_{\pm1.0}$	$89.3_{\pm.5}$	**82.5**$_{\pm.8}$	$86.7_{\pm.4}$	**84.3**$_{\pm.9}$	$86.7_{\pm.6}$
	SOLAR$_{r=4(4\to1.6)}$	82K	**97.0**$_{\pm.5}$	$98.5_{\pm.3}$	$87.9_{\pm.8}$	$91.4_{\pm.4}$	$76.8_{\pm.7}$	$87.1_{\pm.4}$	$78.7_{\pm.7}$	$88.6_{\pm.5}$
	SOLAR$_{r=2(1\to0.3)}$	**50K**	$96.1_{\pm.8}$	$98.2_{\pm.4}$	$87.4_{\pm.9}$	$90.0_{\pm.5}$	$77.0_{\pm.8}$	$86.8_{\pm.6}$	$76.4_{\pm.9}$	$87.6_{\pm.6}$
	SOLAR$_{\text{NOLA}(4\to1.2)}$	64K	$95.8_{\pm.9}$	$97.0_{\pm.4}$	$87.7_{\pm.8}$	$89.3_{\pm.4}$	$82.1_{\pm.7}$	$86.6_{\pm.3}$	$84.1_{\pm.8}$	$86.4_{\pm.6}$

Table 2: Additional evaluation on vision datasets using ViT-B. The table shows bit-level representation footprint (32-bit baseline) and top-1 accuracy. All models are trained for 10 epochs.

Method	Byte Footprint	Oxford Pets	SUN397	CUB-200	ImageNet-1K
LoRA (r=1)	74KB	**93.0**$_{\pm.3}$	**74.3**$_{\pm.2}$	**84.7**$_{\pm.2}$	**81.5**$_{\pm.4}$
NOLA	48KB	$90.4_{\pm.5}$	$61.7_{\pm.4}$	$79.4_{\pm.4}$	$77.4_{\pm.3}$
SOLAR$_{r=1(2\to0.2)}$	**8KB** (89% ↓)	$92.6_{\pm.4}$	$73.9_{\pm.2}$	$84.2_{\pm.3}$	$81.3_{\pm.2}$

Table 3: Effect of quantization on SOLAR$_{r=4(4\to1.6)}$ performance. ViT-L-MAE fine-tuned on CIFAR-10.

Method	Quant.	Accuracy	Byte Footprint
SOLAR	32-bit	$86.7_{\pm.3}$	319KB
	16-bit	$86.5_{\pm.3}$	166KB
	8-bit	$85.9_{\pm.4}$	89KB
	4-bit	$84.8_{\pm.6}$	50KB

Table 4: Effect of rank and adapter placement in SOLAR$_{r=4(4\to1)}$. Accuracy (%) on CIFAR-100 using ViT-B.

Rank	Q	K	V	QV	QKV
1	87.0	85.5	86.6	88.3	90.1
2	87.5	85.7	87.4	88.6	90.5
4	87.8	86.1	87.5	89.0	90.6
8	88.1	86.0	87.4	89.1	90.7
16	87.9	86.0	87.1	89.0	90.6

class) and full-data settings while requiring far fewer trainable parameters than LoRA and NOLA. On ViT-B and ViT-L, SOLAR matches LoRA's performance using up to 74% fewer parameters. For instance, applied to a LoRA ($r = 2$), bases $N_A = N_B = 4000$, and $\text{top}_k = 1600$, SOLAR reduces fine-tuned parameters from 98K to 25K while maintaining comparable accuracy.

Beyond parameter reduction, SOLAR improves storage efficiency. Table 2 reports mean and standard deviation over 5 runs on four additional datasets using ViT-B, quantifying the bit-level footprint assuming 32-bit precision during training. We apply 8-bit quantization to SOLAR after top_k parameter selection. While LoRA ($r = 1$) requires 74KB of adapter parameters, SOLAR reduces this to 8KB (89% reduction). These extreme compressions incur only minor accuracy drops, showing SOLAR enables fine-grained control of model size to meet strict constraints and offers a flexible tradeoff between footprint and performance.

In addition to reducing parameter and storage footprints, SOLAR remains highly robust under quantization. As shown in Table 3, reducing coefficient precision from 32-bit to 4-bit incurs less than a 2% accuracy drop on ViT-L-MAE (CIFAR-10, 10-shot). We further evaluate the effect of adapter rank and placement (Table 4), observing that performance improves with rank up to 8 (with higher ranks requiring more time to converge), and that the Query (Q) projection yields the highest gains.

Table 5: Model representation efficiency for LLaMA models. SOLAR compresses LoRA adapter updates across various model sizes. For the 13B model, all methods use 4-bit quantization, making the LoRA baseline equivalent to QLoRA.

Model	LLaMA-3.2 1B			LLaMA-2 13B (4-bit)		
Method	LoRA r=8	NOLA 1000 bases	SOLAR $r = 8(4 \rightarrow 1.2)$	LoRA r=1	NOLA 1000 bases	SOLAR $r = 1(1 \rightarrow 0.3)$
# Params	852K	**64K**	81K (90% ↓)	819K	140K	**51K** (94% ↓)
Val Loss	**1.51**	1.87	**1.52**	1.05	1.29	**1.05**
MMLU Acc	**30.1**	25.9	28.3	**54.5**	51.8	**54.5**

Table 6: Performance and parameter efficiency on E2E NLG using GPT-2 Small and Medium. All methods use rank-4 adapters applied to the Query and Value projections.

Method	GPT-2 Small		GPT-2 Medium	
	MET	# Params	MET	# Params
Full-FT	28.4	124M	46.2	355M
LoRA (r=4)	**29.7**	147K	**47.2**	393K
NOLA	29.1	48K	46.8	350K
SOLAR (r=4, $1 \rightarrow 0.3$)	**29.7**	15K (90% ↓)	46.4	30K (92% ↓)
SOLAR (r=1, $0.1 \rightarrow 0.1$)	26.1	**4K** (97% ↓)	44.8	**9K** (98% ↓)

3.2 SOLAR ON LLAMA

Experimental Setup. We apply SOLAR to LLaMA-3 models of size 1B–13B. All models are fine-tuned using adapters in the query and value projections across all transformer layers. For the 1B model, we use LoRA with rank 8; for the 31B model, we use LoRA with rank 1. To reduce GPU memory usage for large-scale models, we quantize the 13B model using 4-bit NF4 quantization through the `BitsAndBytes` library Dettmers et al. (2021); Dettmers (2025). Further implementation details and hardware configurations are provided in the Appendix.

Evaluation Benchmarks. All models are fine-tuned on the Stanford Alpaca Taori et al. (2023) dataset for instruction-following and evaluated on its validation loss. We also assess generalization to out-of-distribution tasks using the MMLU benchmark Hendrycks et al. (2020).

Comparison Methods. We compare SOLAR with PEFT baselines, including LoRA Hu et al. (2021) and NOLA Koohpayegani et al. (2024). LoRA uses rank $r = 8$ for LLaMA-3 1B and $r = 1$ for the 13B model. NOLA follows its original configuration, with 1000 random basis vectors per matrix Koohpayegani et al. (2024). For the 13B model, we apply 4-bit quantization to all methods (LoRA, NOLA, and SOLAR). The reported trainable parameters include learned coefficients and overhead for basis indexing. All experiments use gradient checkpointing, and learning rates are tuned separately per model and method to ensure a fair comparison.

Results and Performance Analysis. Table 5 reports results across model sizes. SOLAR matches LoRA in Alpaca validation loss and MMLU Hendrycks et al. (2020) accuracy while reducing trainable adapter parameters by up to 94%. For example, on LLaMA-3.2 13B, SOLAR cuts the adapter size from 819K to 51K without accuracy loss.

3.3 SOLAR ON GPT-2

Experimental Setup. We evaluate our method on GPT-2 Radford et al. (2019) base and medium models fine-tuned on the E2E NLG dataset Novikova et al. (2017) using LoRA. The models are trained for 5 epochs using a batch size of 8 and a learning rate of 0.1. LoRA is applied to the self-attention Query and Value projection, with a rank of $r = 4$. After training, we apply SOLAR to compress the LoRA adapter updates.

Evaluation Benchmarks. We use the E2E NLG dataset to evaluate generative quality. Generated outputs are assessed using METEOR Banerjee & Lavie (2005) metric. We report LoRA, NOLA, and SOLAR performance.

Results and Performance Analysis. Table 6 summarizes results on the E2E NLG dataset using GPT-2 Small and Medium models. SOLAR achieves competitive METEOR scores compared to LoRA and NOLA, while substantially reducing adapter size. On GPT-2 Medium, SOLAR reduces adapter representation size from 393K (LoRA) to 30K parameters with minimal performance loss. Applied to rank-1 LoRA, it achieves a 98% reduction, demonstrating strong compression capability.

3.4 Discussion and Analysis on SOLAR Performance and Efficiency

Subspace Analysis. We analyze the subspace similarity between the foundation model's weights W and the LoRA update ΔW with rank $r = 4$ (see Figure 2). Let $W = U_W \Sigma_W V_W^\top$ and $\Delta W = U_{\Delta W} \Sigma_{\Delta W} V_{\Delta W}^\top$ denote their SVDs. To quantify subspace alignment, we define the similarity function as $\phi(W, \Delta W, i, j) = \psi(U_W^{(i)}, U_{\Delta W}^{(j)}) = \|U_W^{(i)\top} U_{\Delta W}^{(j)}\|_F^2$, where $U_W^{(i)}$ and $U_{\Delta W}^{(j)}$ are the matrices formed by taking the i and j left singular vectors of W and ΔW, respectively. This normalized Frobenius inner product measures how much of the j-dimensional subspace of ΔW lies within the i-dimensional subspace of W, reaching its maximum when perfectly aligned. Figure 2 shows that the fine-tuned model emphasizes directions already present in the foundation model, supporting prior observations that LoRA updates lie in low-dimensional, structured subspaces Hu et al. (2021); Farhadzadeh et al. (2025); Zhang et al. (2025b). This suggests leveraging existing directions is more effective than relying purely on random ones: LoRA implicitly aligns with them, and SOLAR exploits this alignment in its basis pool, explaining its performance advantage over NOLA.

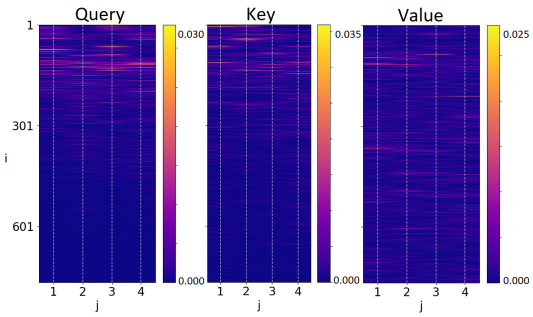

Figure 2: Subspace similarity between the W and ΔW matrices (Q, K, V) from the first layer of the ViT-B model using LoRA with rank $r = 4$.

Effect of Basis Pool Size and Communication Budget on Performance. To evaluate SOLAR's trade-off between representation size and performance, we analyze the effect of varying the basis pool size and the number of selected top$_k$ components on representation accuracy. Experiments are conducted on a ViT-Base model fine-tuned using LoRA with rank 4, followed by SOLAR compression. Each LoRA matrix A and B requires $4 \times 768 = 3072$ parameters. We observe that increasing k improves SOLAR's expressiveness and accuracy. Moreover, a larger basis pool enhances performance by increasing the likelihood of capturing directions aligned with the fine-tuned model subspace. As shown in Figure 3, even with fixed k, larger pools yield higher accuracy by enabling more precise reconstruction of target directions. SOLAR thus achieves performance comparable to LoRA with significantly fewer parameters.

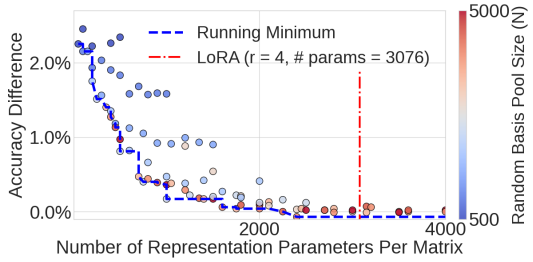

Figure 3: Representation Performance vs. Representation Cost: On ViT-B ($r = 4$), SOLAR demonstrates a trade-off between parameter count and performance, achieving strong performance with far fewer parameters than LoRA.

SOLAR Overhead and Runtime Efficiency. As a post-training method, SOLAR introduces negligible runtime overhead and does not interfere with fine-tuning. For instance, fine-tuning LLaMA-3.2 1B with LoRA on Tiny-ImageNet took 2081 seconds, while SOLAR, including random basis generation, convex least-squares solving, and top$_k$ selection, took only 15 seconds (under 0.72% of training time). These operations are computationally lightweight, as shown in Table 7, confirming SOLAR's practical efficiency.

Table 7: Runtime Overhead: LoRA (10 epochs) vs. SOLAR post-training on ViT-B across vision datasets. Times in seconds.

Dataset	LoRA	SOLAR	Overhead (%)
CIFAR-10	1176	14	1.19
CIFAR-100	1165	14	1.20
Food-101	3480	67	1.92
Tiny-ImageNet	2081	15	0.72
ImageNet-1K	56634	155	0.27

Limitations and Future Work. As a post-hoc method, SOLAR's performance is limited by the base adapter, and its hyperparameters (N and k) may need per-task tuning to optimize the compression-accuracy trade-off. While it shows strong results on vision and language tasks, its effectiveness on other modalities (audio, time series, or multimodal data) remains untested. Future work will extend SOLAR to these areas and evaluate its performance in other environments.

4 Background and Related Works

Transformers in NLP and Vision. Transformers Vaswani et al. (2017), are now the standard in NLP for modeling long-range dependencies via self-attention Raiaan et al. (2024). Models such as LLaMA Touvron et al. (2023), BERT Devlin et al. (2019), and GPT Radford et al. (2018) build on this structure to achieve strong results across diverse benchmarks. In vision, ViT Dosovitskiy et al. (2020) treats image patches as tokens, making Transformers a unifying backbone across modalities.

Parameter-Efficient Fine-Tuning (PEFT). As transformers scale, task-specific fine-tuning becomes computationally intensive. PEFT methods mitigate this by updating only a subset of parameters. LoRA Hu et al. (2021) introduces trainable low-rank matrices per layer, typically modifying <1% of weights, while NOLA Koohpayegani et al. (2024) re-parameterizes these as linear combinations of random bases, decoupling parameters from rank and architecture. Yet PEFT gains often fall short in deployment, especially on edge, mobile, and federated settings with communication and storage bottlenecks. Adapting GPT-2 (117M) on-device may still require gigabytes of transfer and petaflop-scale computation per round Wang et al. (2025), with updates taking seconds to transmit and hours to process on low-power hardware (e.g., Jetson TX2).

Challenges of PEFT. As models grow, adapter overhead scales rapidly. Even modest adapters (e.g., 7M parameters for a 7B model at rank 16) accumulate significant costs across users, tasks, or training rounds Xu et al. (2023b). A 1% adapter for LLaMA-2 70B adds 700M parameters; for GPT-3 (350B), 3.5B—tens of gigabytes in FP32. Such costs are infeasible in personalized or federated settings, where hundreds of adapters may be exchanged or stored per user Zhang et al. (2024). While PEFT leverages the low intrinsic dimensionality of task adaptation Hu et al. (2021), deployment remains inefficient. It has been shown that BERT fine-tuning on MRPC Dolan & Brockett (2005) requires only 1,861 degrees of freedom out of 110M, highlighting redundancy in full-rank updates Aghajanyan et al. (2020). Yet even small adapters impose substantial overhead on massive models Xu et al. (2023a); Lialin et al. (2023). Hence, the true bottleneck is adapter size, not fine-tuning efficiency Jie et al. (2023), motivating flexible post-training compression to reduce footprint without altering training.

PEFT Compression Techniques. To mitigate PEFT costs, pruning Han et al. (2024); Ilhan et al. (2024) and quantization Chen et al. (2024); Hubara et al. (2021) have been explored. These reduce model size but require careful tuning or retraining, are less effective under severe bandwidth limits, and are mainly optimized for full-model compression, limiting applicability to adapters. Adapter updates are highly redundant and lie in low-dimensional subspaces Hu et al. (2021); Yadav et al. (2023); Wu et al. (2024), motivating post-training compression. Methods like ComPEFT Yadav et al. (2023), BitDelta Liu et al. (2024), Delta-CoMe Ping et al. (2024), and DeltaZip Yao et al. (2025) compress adapter weights after fine-tuning but rely on heuristics, task-specific tuning, or training integration, reducing flexibility. Other approaches alter fine-tuning itself: VeRA Kopiczko et al. (2023) employs a shared random basis, SVFT Lingam et al. (2024) learns sparse coefficients for an SVD-based basis, and EigenLoRAx Kaushik et al. (2025) builds a PCA basis from many pre-trained adapters. In contrast, SOLAR is a post-hoc, training-free utility that compresses any adapter, providing a complementary plug-and-play solution.

5 Conclusion

Adapter-based fine-tuning methods such as LoRA significantly reduce the cost of adapting large models. However, in distributed and on-device settings, communication and storage overheads remain a major bottleneck. To address this, we introduce SOLAR, a lightweight post-training compression method that reparameterizes adapter updates as sparse combinations of structured basis vectors aligned with the foundation model's latent subspace. SOLAR substantially reduces adapter size and transmission cost without altering the training process or model architecture.

REFERENCES

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pp. 446–461. Springer, 2014.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Tim Dettmers. Bitsandbytes: 8-bit optimizers and quantization. `https://github.com/TimDettmers/bitsandbytes`, 2025. Accessed: 15-May-2025.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Farzad Farhadzadeh, Debasmit Das, Shubhankar Borse, and Fatih Porikli. Lora-x: Bridging foundation models with training-free cross-model adaptation. *arXiv preprint arXiv:2501.16559*, 2025.

Chao Gao and Sai Qian Zhang. Dlora: Distributed parameter-efficient fine-tuning solution for large language model. *arXiv preprint arXiv:2404.05182*, 2024.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2): 217–288, 2011.

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.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pp. 4466–4475. PMLR, 2021.

Fatih Ilhan, Gong Su, Selim Furkan Tekin, Tiansheng Huang, Sihao Hu, and Ling Liu. Resource-efficient transformer pruning for finetuning of large models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16206–16215, 2024.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Shibo Jie, Haoqing Wang, and Zhi-Hong Deng. Revisiting the parameter efficiency of adapters from the perspective of precision redundancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17217–17226, 2023.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021.

Prakhar Kaushik, Ankit Vaidya, Shravan Chaudhari, and Alan Yuille. Eigenlorax: Recycling adapters to find principal subspaces for resource-efficient adaptation and inference. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 649–659, 2025.

Soroush Abbasi Koohpayegani, KL Navaneet, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. Nola: Compressing lora using linear combination of random basis. *ICLR 2024*, 2024.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

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

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 2023.

Vijay Chandra Lingam, Atula Neerkaje, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Eunsol Choi, Alex Dimakis, Aleksandar Bojchevski, and Sujay Sanghavi. Svft: Parameter-efficient fine-tuning with singular vectors. *Advances in Neural Information Processing Systems*, 37:41425–41446, 2024.

James Liu, Guangxuan Xiao, Kai Li, Jason D Lee, Song Han, Tri Dao, and Tianle Cai. Bitdelta: Your fine-tune may only be worth one bit. *Advances in Neural Information Processing Systems*, 37: 13579–13600, 2024.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, et al. Parameter-efficient orthogonal finetuning via butterfly factorization. *arXiv preprint arXiv:2311.06243*, 2023.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pp. 23610–23641. PMLR, 2023.

Per-Gunnar Martinsson and Joel A Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.

Elissa Mhanna and Mohamad Assaad. Countering the communication bottleneck in federated learning: A highly efficient zero-order optimization technique. *Journal of Machine Learning Research*, 25(418):1–53, 2024.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*, 2017.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

A Paszke. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Bowen Ping, Shuo Wang, Hanqing Wang, Xu Han, Yuzhuang Xu, Yukun Yan, Yun Chen, Baobao Chang, Zhiyuan Liu, and Maosong Sun. Delta-come: Training-free delta-compression with mixed-precision for large language models. *arXiv preprint arXiv:2406.08903*, 2024.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems*, 36:79320–79362, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Mohaimenul Azam Khan Raiaan, Md Saddam Hossain Mukta, Kaniz Fatema, Nur Mohammad Fahad, Sadman Sakib, Most Marufatul Jannat Mim, Jubaer Ahmad, Mohammed Eunus Ali, and Sami Azam. A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE access*, 12:26839–26874, 2024.

Google Research. Vision Transformer Models on Hugging Face. `https://huggingface.co/google`, 2025. Accessed: 06-May-2025.

Mariia Seleznova, Dana Weitzner, Raja Giryes, Gitta Kutyniok, and Hung-Hsu Chou. Neural (tangent kernel) collapse. *Advances in Neural Information Processing Systems*, 36:16240–16270, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Shilong Wang, Jianchun Liu, Hongli Xu, Jiaming Yan, and Xianjun Gao. Efficient federated fine-tuning of large language models with layer dropout. *arXiv preprint arXiv:2503.10217*, 2025.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

Ross Wightman. timm: PyTorch Image Models. `https://github.com/huggingface/pytorch-image-models/tree/main/timm`, 2025. Accessed: 06-May-2025.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. *arXiv preprint arXiv:2406.11909*, 2024.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023a.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*, 2023b.

Prateek Yadav, Leshem Choshen, Colin Raffel, and Mohit Bansal. Compeft: Compression for communicating parameter efficient updates via sparsification and quantization. *arXiv preprint arXiv:2311.13171*, 2023.

Xiaozhe Yao, Qinghao Hu, and Ana Klimovic. Deltazip: Efficient serving of multiple full-model-tuned llms. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 110–127, 2025.

Chunxu Zhang, Guodong Long, Tianyi Zhou, Zijian Zhang, Peng Yan, and Bo Yang. When federated recommendation meets cold-start problem: Separating item attributes and user interactions. In *Proceedings of the ACM Web Conference 2024*, pp. 3632–3642, 2024.

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. Lora-one: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. *arXiv preprint arXiv:2502.01235*, 2025a.

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. One-step full gradient suffices for low-rank fine-tuning, provably and efficiently. *arXiv preprint arXiv:2502.01235*, 2025b.

APPENDIX

A PROOF OF THEOREM 1

Let $\Delta W^* \in \mathbb{R}^{m\times n}$ denote the optimal adapter for the downstream task, ΔW the adapter obtained by LoRA fine-tuning, and $\Delta\widetilde{W}$ the SOLAR reconstruction. Let ΔW_{proj} denote the projection of ΔW onto the SOLAR bases (i.e., bases that are constructed from the SVD of the foundation model's weights, combined with randomized perturbations).

Our proof relies on the following standard assumptions from the literature on parameter-efficient fine-tuning and randomized numerical linear algebra:

- (A1) *Spectral Initialization:* The LoRA adapter matrices A and B are initialized using the spectral initialization strategy from Zhang et al. (2025a).
- (A2) *Low-Rank Update:* The optimal task-specific update ΔW^* is approximately low-rank, with rank $r^* < \min\{m, n\}$ Zhang et al. (2025a).
- (A3) *Well-Behaved Data:* The training data follows the generation process outlined in Zhang et al. (2025a), where input features are drawn from an isotropic sub-Gaussian or Gaussian distribution.
- (A4) *Fast Spectrum Decay:* The projected update matrix ΔW_{proj} exhibits spectral decay, meaning its tail singular values are small (Martinsson & Tropp, 2020).

First, we decompose the total error using the triangle inequality. The total error, $\|\Delta\tilde{W} - \Delta W^*\|_F$, is the distance between the SOLAR-reconstructed adapter and the optimal adapter. This is bounded by the sum of the Training Error and the Compression Error:

$$\|\Delta\tilde{W} - \Delta W^*\|_F \leq \underbrace{\|\Delta\tilde{W} - \Delta W\|_F}_{\text{Compression Error}} + \underbrace{\|\Delta W - \Delta W^*\|_F}_{\text{Training Error}} \tag{6}$$

Here, the first term, $\|\Delta\tilde{W} - \Delta W\|_F$, is the compression error introduced by SOLAR's approximation. The second term, $\|\Delta W - \Delta W^*\|_F$, is the training error from the underlying LoRA fine-tuning process itself. We will bound each term separately.

The analysis of the training error for LoRA adapters is non-trivial and has been extensively studied. We directly leverage the results from Zhang et al. (2025a), showing that under Assumptions (A1)-(A3), LoRA trained with gradient descent converges to the optimal low-rank adapter ΔW^*. Their analysis provides the following bound on the training error after t steps:

$$\|\Delta W - \Delta W^*\|_F \leq \sqrt{2r^*}\left(1 - \frac{\eta\lambda_{r^*}}{64\kappa}\right)^t \lambda_{r^*}, \tag{7}$$

where r^* is the rank of the optimal update ΔW^*, κ is its condition number, λ_{r^*} is its r^*-th singular value, and η is the learning rate. This bound, derived under the specified spectral initialization and data concentration assumptions, demonstrates that the fine-tuned adapter ΔW gets exponentially closer to the optimal adapter ΔW^* as training progresses.

The compression error arises from SOLAR's two-stage approximation process: 1) projecting the update onto a finite basis pool of size N, and 2) sparsifying the representation to only k non-zero coefficients. We can further decompose this error:

$$\|\Delta\tilde{W} - \Delta W\|_F \leq \underbrace{\|\mathcal{P}_N(\Delta W) - \Delta W\|_F}_{\text{Projection Error}} + \underbrace{\|\Delta\tilde{W} - \mathcal{P}_N(\Delta W)\|_F}_{\text{Sparsification Error}} \tag{8}$$

where $\mathcal{P}_N(\Delta W)$ represents the best possible approximation of ΔW using the full basis set of size N.

SOLAR reconstructs $\Delta\tilde{W}$ by sparsely combining perturbed basis matrices:

$$\Delta\tilde{W} = \sum_i \sum_j \beta_i\alpha_j \left(M_B^{(i)} + \epsilon_i\right)\left(M_A^{(j)} + \epsilon_j\right), \tag{9}$$

with $M_B^{(i)}, M_A^{(j)}$ basis matrices, α_j, β_i sparse coefficients, and $\epsilon_i, \epsilon_j \sim \mathcal{N}(0,1)$.

Expanding the product yields four types of terms:

$$\Delta\tilde{W} = \underbrace{\sum_{i,j} \beta_i \alpha_j M_B^{(i)} M_A^{(j)}}_{\text{dominant term}} + \underbrace{\sum_{i,j} \beta_i \alpha_j \epsilon_i M_A^{(j)}}_{\text{linear in } \epsilon_B} + \underbrace{\sum_{i,j} \beta_i \alpha_j M_B^{(i)} \epsilon_j}_{\text{linear in } \epsilon_A} + \underbrace{\sum_{i,j} \beta_i \alpha_j \epsilon_i \epsilon_j}_{\text{quadratic in } \epsilon}. \tag{10}$$

Because $\mathbb{E}[\epsilon_i] = 0$, the linear terms vanish in expectation:

$$\mathbb{E}\left[\sum_{i,j} \beta_i \alpha_j \epsilon_i M_A^{(j)}\right] = 0, \quad \mathbb{E}\left[\sum_{i,j} \beta_i \alpha_j M_B^{(i)} \epsilon_j\right] = 0. \tag{11}$$

The quadratic term can be bounded using the Cauchy–Schwarz inequality. Let $\epsilon \in \mathbb{R}^{m\times n}$ be a random matrix with i.i.d. entries $\epsilon_{pq} \sim \mathcal{N}(0,1)$, and let α_j, β_i denote the sparse coefficients. Then

$$\mathbb{E}\Big\| \sum_{i,j} \beta_i \alpha_j \epsilon_i \epsilon_j \Big\|_F = \mathbb{E}\left[\Big(\sum_{p,q} \Big| \sum_{i,j} \beta_i \alpha_j \epsilon_i^p \epsilon_j^q \Big|^2 \Big)^{1/2} \right] \tag{12}$$

$$\leq \left(\sum_i |\beta_i|^2 \, \mathbb{E}\|\epsilon_i\|_F^2 \right)^{1/2} \left(\sum_j |\alpha_j|^2 \, \mathbb{E}\|\epsilon_j\|_F^2 \right)^{1/2} \tag{13}$$

$$= \|\beta\|_2 \, \|\alpha\|_2 \, \sqrt{mn}, \quad \text{since } \mathrm{Var}(\epsilon_{pq}) = 1, \tag{14}$$

$$\tag{15}$$

The dominant term $\sum_{i,j} \beta_i \alpha_j M_B^{(i)} M_A^{(j)}$ corresponds to projecting ΔW onto the basis pool of size N. To make the coefficients α, β explicit and account for matrix dimensions, we model this projection using a standard RNLA (randomized numerical linear algebra) rangefinder:

$$Y = \Delta W \, \Omega, \qquad Q = \mathrm{orth}(Y), \qquad \mathcal{P}_N(\Delta W) = QQ^\top \Delta W, \tag{16}$$

where $\Omega \in \mathbb{R}^{n\times N}$ is a random test matrix encoding the selection of basis vectors, and orthonormalization is performed on the columns of Y to obtain $Q \in \mathbb{R}^{m\times N}$.

Under standard RNLA assumptions (Gaussian or sub-Gaussian Ω, oversampling $N - r \geq 2$, and mild moment conditions), the expected Frobenius-norm residual satisfies (Theorem 10.5 of Halko et al. (2011)):

$$\mathbb{E}_\Omega \big\| (I - QQ^\top) \Delta W \big\|_F \;\leq\; \sqrt{1 + \frac{r}{N - r - 1}} \left(\sum_{i=r+1}^{\min(m,n)} \sigma_i^2(\Delta W) \right)^{1/2}, \tag{17}$$

where r is the effective rank of ΔW and the sum over singular values runs up to $\min(m,n)$ to account for the full matrix size.

Including the quadratic-noise term from the Gaussian perturbations in SOLAR, and explicitly accounting for the coefficients α, β and matrix size, we obtain the expected reconstruction error:

$$\mathbb{E}\big\| \Delta\tilde{W} - \Delta W \big\|_F \;\leq\; \underbrace{\mathbb{E}\|\mathcal{P}_N(\Delta W) - \Delta W\|_F}_{\text{projection error}} + \underbrace{\sqrt{mn}\, \|\beta\|_2 \|\alpha\|_2}_{\text{quadratic-noise contribution}}, \tag{18}$$

ensuring that the α, β coefficients explicitly appear in the compression error.

The second step of SOLAR is coefficient sparsification, which selects the top k most significant basis vectors. This corresponds to the best k-term approximation of the projected adapter. Denoting the singular values of $\mathcal{P}_N(\Delta W_{\mathrm{proj}})$ by $\tilde{\sigma}_j$, the sparsification error is bounded as

$$\|\Delta\tilde{W} - \mathcal{P}_N(\Delta W)\|_F \leq \sqrt{\sum_{j=k+1} \tilde{\sigma}_j^2}. \tag{19}$$

Combining projection, sparsification, and quadratic-noise contributions, the total compression error (C_2) is

$$C_2 = \underbrace{\sqrt{1+\frac{r}{N-r-1}}\Big(\sum_{i=r+1}^{\min(m,n)}\sigma_i^2(\Delta W_{\text{proj}})\Big)^{1/2}}_{\text{projection}} + \underbrace{\sqrt{\sum_{j=k+1}\tilde{\sigma}_j^2}}_{\text{sparsification}} + \underbrace{\sqrt{mn}\,\|\beta\|_2\|\alpha\|_2}_{\text{quadratic-noise}}. \quad (20)$$

Finally, combining the fine-tuning error (C_1) with the total compression error (C_2) gives the full reconstruction error bound:

$$\mathbb{E}\big[\|\Delta\tilde{W} - \Delta W^*\|_F\big] \le \underbrace{\sqrt{2r^*}\left(1-\frac{\eta\lambda_{r^*}}{64\kappa}\right)^t \lambda_{r^*}}_{C_1} + C_2, \quad (21)$$

where C_1 depends on the fine-tuning dynamics (learning rate, number of steps, spectrum of ΔW^*), and C_2 explicitly accounts for SOLAR's compression.

B Implementation Details

All models are implemented using PyTorch Paszke (2019), with HuggingFace Transformers Wolf et al. (2020) for LLaMA and GPT-based models, and Timm Wightman (2025) for ViT-based vision backbones. Training and evaluation are performed on NVIDIA A100 and RTX 4090 GPUs. For all vision experiments, we use ViT-B and ViT-L as base encoders. For language models, we use GPT-2 and LLaMA-3 (1B, 3B, 8B). LoRA is applied to the query and value projections. SOLAR operates post-training by compressing the PEFT adapter matrices. All experiments are conducted under a fixed random seed for reproducibility. The implementation code for SOLAR, along with scripts used to reproduce the experiments, is included in the supplementary material and also available at `https://anonymous.4open.science/r/SOLAR-D3B2/`.

C Dataset Details

We summarize dataset statistics in Table 8, including number of training samples and class counts.

Table 8: Dataset statistics used in experiments. Each dataset includes the number of training samples and classes.

Dataset	Training Samples	Number of Classes
CIFAR-10	50,000	10
CIFAR-100	50,000	100
Food-101	75,750	101
Tiny-ImageNet	100,000	200
ImageNet-1K	1,281,167	1,000

We summarize dataset statistics used in the LLM experiments in Table 9, covering instruction tuning (Section 3.2) and language generation tasks (Section 3.3). The table includes the number of training samples, average sequence lengths, and the model-specific context in which each dataset is used in the experiments.

D Representation Cost Details: Parameters and Storage

To quantify SOLAR's compression benefit, we detail the number of adapter parameters and byte-level footprint across ViT-B, ViT-L, LLaMA, and GPT-2 models. We compare LoRA, NOLA, and SOLAR under adapter rank ($r = 4$). Tables 10 through 15 provide full parameter breakdowns. Byte-level analysis is presented in Table 13.

Table 9: Dataset statistics in LLM experiments.

Dataset	Samples	Avg. Seq. Length	Context
Stanford Alpaca	52,000	∼256 tokens	LLaMA-3 instruction tuning
MMLU	15,858	∼200 tokens	LLaMA-3 Generalization evaluation
E2E NLG	42,000	∼35 tokens	GPT-2 generation fine-tuning

ViT. For vision backbones, Table 10 and Table 11 report the number of representation parameters for query projections (Q) and classifier heads. In the experiments presented in the main paper, the classifier head parameters are excluded from comparison since they are identical across all methods following Koohpayegani et al. (2024). NOLA's parameter footprint for MLP projections is shown in Table 12 (following the setup in Koohpayegani et al. (2024)). Byte-level storage comparisons across quantization, used to produce Table 2 and Table 3 in the main paper, are provided in Table 13.

Table 10: Number of representation parameters for ViT-B (Rank = 4). Each row reports the parameter count for query projections and the classifier head using SOLAR and LoRA across different datasets. The classifier head parameter count is shared across methods and is computed as `(num_classes × 768 + num_classes)`. For SOLAR, the query projection count corresponds to: number of layers × (top$_k$ coefficients for A + top$_k$ coefficients for B + encoded basis for A + encoded basis for B) +1 (seed value). All SOLAR rows follow the form $N \rightarrow \text{top}_k$ where N is the original subspace size. For LoRA, the query projection count corresponds to: number of layers × (input dimension × rank for A + rank × output dimension for B), where rank is 4.

Method	Dataset	Query (Q)	Classifier Head
SOLAR	CIFAR-10	$12 \times \left((1600 + 1600) + \frac{4000+4000}{32}\right) + 1 = 41{,}401$	$10 \times 768 + 10 = 7{,}690$
	CIFAR-100	41,401	$100 \times 768 + 100 = 76{,}900$
	Food-101	41,401	$101 \times 768 + 101 = 77{,}669$
	Tiny-ImageNet	41,401	$200 \times 768 + 200 = 154{,}000$
LoRA	CIFAR-10	$12 \times [(768 \times 4) + (4 \times 768)] = 73{,}728$	$10 \times 768 + 10 = 7{,}690$
	CIFAR-100	73,728	$100 \times 768 + 100 = 76{,}900$
	Food-101	73,728	$101 \times 768 + 101 = 77{,}669$
	Tiny-ImageNet	73,728	$200 \times 768 + 200 = 154{,}000$

Table 11: Number of representation parameters for ViT-L (Rank = 4). Each row shows the parameter counts for Query projections and the classifier head using SOLAR and LoRA across different datasets. The classifier head parameter count is shared across methods and is calculated as `(num_classes × 1024 + num_classes)`.

Method	Dataset	Query (Q)	Classifier Head
SOLAR	CIFAR-10	$24 \times \left((500 + 500) + \frac{1000+1000}{32}\right) + 1 = 25{,}501$	$10 \times 1024 + 10 = 10{,}250$
	CIFAR-100	25,501	$100 \times 1024 + 100 = 102{,}500$
	Food-101	25,501	$101 \times 1024 + 101 = 103{,}625$
	Tiny-ImageNet	25,501	$200 \times 1024 + 200 = 204{,}800$
LoRA	CIFAR-10	$24 \times [(1024 \times 4) + (4 \times 1024)] = 196{,}608$	$10 \times 1024 + 10 = 10{,}250$
	CIFAR-100	196,608	$100 \times 1024 + 100 = 102{,}500$
	Food-101	196,608	$101 \times 1024 + 101 = 103{,}625$
	Tiny-ImageNet	196,608	$200 \times 1024 + 200 = 204{,}800$

LLMs. For language models, parameter counts for adapter layers are detailed in Table 14 for LLaMA and in Table 15 for GPT-2 variants.

E Additional Experimental Results

This section provides supplementary experimental results to further validate the claims made in the main paper. We present detailed performance metrics for additional model scales and include a crucial ablation study that compares SOLAR against a parameter-matched LoRA baseline.

Table 12: Number of representation parameters for ViT-B (Rank = 4). Each row shows the parameter counts for MLP projections (for NOLA) and classifier head across datasets. The classifier head parameter count is shared across methods and is calculated as `(num_classes × 768 + num_classes)`.

Method	Dataset	MLP	Classifier Head
NOLA	CIFAR-10	$12 \times 2 \times 2 \times 1000 + 1 = 48{,}001$	$10 \times 768 + 10 = 7{,}690$
	CIFAR-100	48,001	$100 \times 768 + 100 = 76{,}900$
	Food-101	48,001	$101 \times 768 + 101 = 77{,}669$
	Tiny-ImageNet	48,001	$200 \times 768 + 200 = 154{,}000$

Table 13: Byte-level footprint of representation parameters for ViT-B and ViT-L using LoRA and SOLAR. Each value reflects the total number of bytes required to store adapter updates (excluding classifier heads). For LoRA, storage is computed as: number of layers $\times$ $\big($rank $\times$ output dimension for B + input dimension $\times$ rank for $A\big)$ $\times$ precision in bytes (e.g., 4 bytes for 32-bit float). For SOLAR, storage is computed as: number of layers $\times$ $\big($top$_k$ coefficients for A + top$_k$ coefficients for B + encoded basis vectors for A + encoded basis for $B\big)$ $\times$ precision in bytes, plus 1 byte to store a random seed. For example, the row "$500 \to 50$" denotes that 500-dimensional subspaces are sparsified to top-$k = 50$ coefficients, with encoded bases represented at 1 bit per element (8 elements per byte).

Method	Representation Footprint (Bytes)
LoRA (r=1)	$12 \times [(768 \times 1) + (1 \times 768)] \times 4 = 73{,}728$
SOLAR for ViT-B 8Bit (r=1, $500 \to 50$)	$12 \times \left[(50 + 50) + \frac{500}{8}\right] \times 1 + 1 = 1{,}951$
SOLAR for ViT-B 8Bit (r=1, $100 \to 10$)	$12 \times \left[(10 + 10) + \frac{100}{8}\right] \times 1 + 1 = 391$
LoRA (r=4)	$24 \times [(1024 \times 4) + (4 \times 1024)] \times 4 = 786{,}432$
SOLAR for ViT-L 32Bit (r=4, $4000 \to 1600$)	$24 \times \left[(1600 + 1600) + \frac{4000}{32}\right] \times 4 + 1 = 319{,}201$
SOLAR for ViT-L 16Bit (r=4, $4000 \to 1600$)	$24 \times \left[(1600 + 1600) + \frac{4000}{16}\right] \times 2 + 1 = 165{,}601$
SOLAR for ViT-L 8Bit (r=4, $4000 \to 1600$)	$24 \times \left[(1600 + 1600) + \frac{4000}{8}\right] \times 1 + 1 = 88{,}801$
SOLAR for ViT-L 4Bit (r=4, $4000 \to 1600$)	$24 \times \left[(1600 + 1600) + \frac{4000}{4}\right] \times 0.5 + 1 = 50{,}401$

E.1 Performance on Intermediate-Scale LLaMA Models

Table 16 extends our analysis to the LLaMA-3.2 3B and LLaMA-3.1 8B models, demonstrating SOLAR's consistent efficiency and performance on intermediate-scale architectures. The results show that SOLAR maintains the performance of the original LoRA adapters while achieving parameter reductions of over 90%.

E.2 Extreme Compression

In this section, we report additional experiments demonstrating SOLAR's ability to achieve extreme compression while retaining competitive accuracy. These results complement the main paper by highlighting scenarios where communication and storage constraints are especially strict (e.g., distributed or on-device learning).

Table 17 shows evaluations on four vision datasets using ViT-B under different compression budgets. We quantify the bit-level representation footprint assuming 32-bit precision during training and apply 8-bit quantization to the SOLAR coefficients after top-k selection. Compared to LoRA ($r = 1$), SOLAR reduces the adapter footprint by up to 99% (from 74KB to 0.4KB) with only minor drops in accuracy. These results illustrate that SOLAR enables fine-grained tradeoffs between accuracy and storage cost under extreme compression budgets.

Table 14: Number of representation parameters for LLaMA-3 models using LoRA, NOLA, and SOLAR. Each row reports total adapter parameters for attention projections (Q and V for LoRA and NOLA; Q and K for SOLAR). Output heads and MLP layers are frozen. For LoRA, the parameter count is computed as: number of layers $\times$ $\big($input dimension $\times$ rank for B + rank $\times$ output dimension for A + $\big)$. Due to differing dimensions between A and B in LoRA, the table computes the contributions for Q and V projections separately. For NOLA, it is computed as: number of layers $\times$ 2 $\times$ (number of random basis vectors), assuming separate basis sets for A and B. For SOLAR, the count is: number of layers $\times$ 2 $\times$ $\big($top$_k$ coefficients for B + top$_k$ for A + encoded bases for B + encoded bases for $A\big)$, plus 1 byte to communicate or store the shared seed.

Model (Rank)	Configuration	Total Parameters
LLaMA-3.2 1B (r=8)	16 layers (Q, V)	$16 \times [(2048 \times 8 + 8 \times 2048) + (2048 \times 8 + 8 \times 512)] = 851{,}968$
NOLA	16 layers (Q, V)	$16 \times 2 \times (1000 + 1000) = 64{,}000$
SOLAR (r=8,4K$\rightarrow$1.2K)	16 layers (Q, V)	$16 \times 2 \times \big(1200 + 1200 + \frac{4000}{32}\big) + 1 = 80{,}801$
LLaMA-3.2 3B (r=1)	28 layers (Q, V)	$28 \times [(3072 \times 1 + 1 \times 3072) + (3072 \times 1 + 1 \times 1024)] = 286{,}720$
NOLA	28 layers (Q, V)	$28 \times 2 \times (1000 + 1000) = 112{,}000$
SOLAR (r=1,1000$\rightarrow$150)	28 layers (Q, V)	$28 \times 2 \times \big(150 + 150 + \frac{1000}{32}\big) + 1 = 18{,}551$
LLaMA-3.1 8B (r=1)	32 layers (Q, V)	$32 \times [(4096 \times 1 + 1 \times 4096) + (4096 \times 1 + 1 \times 1024)] = 425{,}984$
NOLA	32 layers (Q, V)	$32 \times 2 \times (1000 + 1000) = 128{,}000$
SOLAR (r=1, 1000$\rightarrow$300)	32 layers (Q, V)	$32 \times 2 \times \big(300 + 300 + \frac{1000}{32}\big) + 1 = 40{,}401$

Table 15: Number of trainable adapter parameters for GPT-2 models using LoRA, NOLA, and SOLAR. Each row reports the total number of parameters added to the query and value projections (Q and V). All configurations freeze the output heads and MLP layers. For LoRA, the parameter count is computed as: number of layers $\times$ 2 $\times$ $\big($input dimension $\times$ rank for B + rank $\times$ output dimension for $A\big)$. For NOLA, the parameter count is: number of layers $\times$ 2 $\times$ (number of random basis vectors), assuming separate basis sets for Q and V. For SOLAR, the parameter count is: number of layers $\times$ 2 $\times$ $\big($top$_k$ coefficients for B + top$_k$ coefficients for A + encoded bases for B + encoded bases for $A\big)$, plus 1 for the shared seed.

Model (Rank)	Configuration	Total Parameters
GPT-2 Small (r=4)	12 layers (Q, V)	$12 \times 2 \times (768 \times 4 + 4 \times 768) = 147{,}456$
NOLA	12 layers (Q, V)	$12 \times 2 \times (1000 + 1000) = 48{,}000$
SOLAR (r=1, 1000$\rightarrow$300)	12 layers (Q, V)	$12 \times 2 \times \big(300 + 300 + \frac{1000}{32}\big) + 1 = 15{,}150$
SOLAR (r=1, 100$\rightarrow$90)	12 layers (Q, V)	$12 \times 2 \times \big(90 + 90 + \frac{100}{32}\big) + 1 = 4{,}396$
GPT-2 Medium (r=4)	24 layers (Q, V)	$24 \times 2 \times (1024 \times 4 + 4 \times 1024) = 393{,}216$
NOLA	24 layers (Q, V)	350,000 Koohpayegani et al. (2024)
SOLAR (r=4, 1000$\rightarrow$300)	24 layers (Q, V)	$24 \times 2 \times \big(300 + 300 + \frac{1000}{32}\big) + 1 = 30{,}301$
SOLAR (r=4, 100$\rightarrow$90)	24 layers (Q, V)	$24 \times 2 \times \big(90 + 90 + \frac{100}{32}\big) + 1 = 8{,}791$

F Scalability to Larger Vision Models

To validate that SOLAR remains effective and computationally tractable on larger-scale models, we conducted experiments on the ViT-G/14 architecture. This model is substantially larger than the ViT-B/L backbones used in our main experiments, providing a strong test of scalability.

We fine-tuned a ViT-G/14 model on the full CIFAR-10, CIFAR-100, Food-101, and T-ImageNet datasets using a LoRA adapter with rank $r = 4$. We then applied SOLAR with a basis pool of 8,000 vectors, selecting the top 4,000 coefficients to form the compressed adapter.

As shown in Table 18, SOLAR successfully preserves the performance of the original LoRA adapter with negligible accuracy drops, while reducing the adapter's parameter count by 31% (from 492K to 340K). This result demonstrates that SOLAR's core mechanisms—including SVD extraction and sparse reconstruction—scale effectively to larger models without sacrificing compression efficiency or task performance.

Table 16: Model representation efficiency for LLaMA 3B and 8B models. For the 8B model, all methods use 4-bit quantization, making the LoRA baseline equivalent to QLoRA.

Model	LLaMA-3.2 3B			LLaMA-3.1 8B (4-bit)		
Method	LoRA r=1	NOLA 1000 bases	SOLAR SOLAR$_{r=1(1K\to0.1K)}$	LoRA r=1	NOLA 1000 bases	SOLAR SOLAR$_{r=1(1K\to0.3K)}$
# Params	287K	112K	**16K** (94% ↓)	425K	128K	**40K** (91% ↓)
Val Loss	**1.02**	1.31	**1.04**	**0.89**	1.01	**0.90**
MMLU Acc	**54.0**	52.7	**54.0**	**60.9**	56.1	**60.9**

Table 17: Evaluation of extreme compression on ViT-B. We report bit-level representation footprint (32-bit baseline) and top-1 accuracy over 5 runs. All models are trained for 10 epochs.

Method	Byte Footprint	Oxford Pets	SUN397	CUB-200	ImageNet-1K
LoRA (r=1)	74KB	**93.0**$_{\pm 0.5}$	**74.3**$_{\pm 0.3}$	**84.7**$_{\pm 0.4}$	**81.5**$_{\pm 0.6}$
SOLAR (r=1, 500→50)	2KB (97% ↓)	91.2$_{\pm 0.6}$	72.4$_{\pm 0.4}$	81.4$_{\pm 0.5}$	80.7$_{\pm 0.4}$
SOLAR (r=1, 100→10)	**0.4KB** (99% ↓)	90.3$_{\pm 0.7}$	72.4$_{\pm 0.5}$	81.3$_{\pm 0.6}$	80.6$_{\pm 0.5}$

Table 18: Scalability of SOLAR on the ViT-G/14 model. Results show top-1 accuracy (%) on full datasets.

Method	# Params	CIFAR-10	CIFAR-100	Food-101	T-ImageNet
LoRA ($r = 4$)	492K	99.4	94.6	91.2	92.8
SOLAR ($r = 4, 8\text{K} \to 4\text{K}$)	340K (31% ↓)	99.4	94.5	91.2	92.8

F.1 Ablation Study: Budget-Matched LoRA Comparison

To further validate the efficiency of our compression strategy, we conduct an ablation study directly comparing SOLAR to a budget-matched LoRA baseline, as suggested by reviewer feedback.[1] This comparison is critical to demonstrate that SOLAR's benefits extend beyond mere parameter reduction and offer a more effective performance-compression trade-off than simply training a lower-rank adapter from scratch.

As shown in Table 19, fine-tuning a LoRA adapter with a reduced rank (r=2) to match the parameter count of the compressed SOLAR adapter results in a significant performance degradation across all tasks. In contrast, SOLAR, when applied to the higher-performing LoRA (r=4) adapter, successfully preserves task accuracy while achieving a comparable parameter budget. This highlights that SOLAR retains the expressive power of the original higher-rank adapter, a feat not achievable by simply reducing the rank during training. All experiments were conducted on the full datasets using the ViT-B backbone, with results reported as the mean accuracy over five independent runs to ensure statistical robustness.

Table 19: Comparison of SOLAR with a budget-matched LoRA (r=2) baseline on ViT-B. While LoRA (r=2) has a similar parameter count to the compressed SOLAR adapter, it shows a clear performance degradation. SOLAR maintains performance comparable to the original, higher-rank LoRA (r=4).

Method	#Params	CIFAR-10	CIFAR-100	Food-101	T-ImageNet
LoRA ($r = 4$)	74K	98.3	90.3	87.6	88.8
LoRA ($r = 2$)	37K	97.1	89.0	85.5	87.4
SOLAR ($r = 4, 4\text{K} \to 1.6\text{K}$)	41K	98.3	89.8	87.0	87.9
SOLAR ($r = 4, 4\text{K} \to 0.8\text{K}$)	22K	97.0	89.0	85.2	87.4

G Application to Federated Learning

One of the motivations for developing SOLAR is to reduce communication overhead in distributed learning scenarios, such as Federated Learning (FL). In typical FL setups, clients fine-tune a model on their local data and transmit the resulting model updates (e.g., LoRA adapters) to a central server for aggregation. As highlighted by recent work Mhanna & Assaad (2024), communication—not computation—is often the primary bottleneck. Transmitting full adapters from thousands of clients can generate enormous data transfer loads. For example, in an FL setup with 10,000 clients—1,000 participating in each of 10 training rounds—transmitting 74 KB LoRA adapters per client would amount to 740 GB of total data transfer.

SOLAR addresses this challenge as a lightweight, post-hoc compression utility. After local training, each client can compress its adapter with SOLAR before transmission. The server then receives only the sparse coefficients and a random seed, drastically reducing per-client communication costs.

To demonstrate SOLAR's effectiveness in distributed settings, we simulated a 10-client FL environment. We compare a baseline where clients transmit full LoRA adapters with a scenario where clients transmit SOLAR-compressed adapters. Each client fine-tunes a ViT-B model on CIFAR-10 with LoRA ($r = 4$), under two data distribution scenarios: an IID baseline and a non-IID distribution generated via a Dirichlet process with a concentration parameter of 0.5. The simulation runs for 30 communication rounds, with one epoch of local training per client per round.

As shown in Table 20, the performance gap between full LoRA adapters and SOLAR-compressed adapters is minimal in both IID and non-IID settings. This demonstrates that SOLAR's compression does not disproportionately harm aggregation performance, even under significant data heterogeneity. Our experiment confirms that SOLAR can serve as a post-training, plug-and-play module to reduce communication costs in standard FL frameworks without requiring complex changes to the aggregation strategy.

Table 20: Performance of SOLAR on ViT-B under IID and non-IID data distributions in a simulated 10-client federated learning environment.

Method	# Params	CIFAR-10 (IID)	CIFAR-10 (non-IID)
LoRA ($r = 4$)	74K	93.7	87.4
SOLAR ($r = 4, 4\text{K} \rightarrow 2\text{K}$)	51K (31% $\downarrow$)	93.2	86.7