PARAMETER-EFFICIENT SUBSPACE OPTIMIZATION FOR LLM FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper develops a new perspective on parameter-efficient training for LLMs, inspired by the classical theory of subspace minimization. We introduce a unifying framework, **P**arameter-**E**fficient **S**ubspace **O**ptimization (**PESO**), which not only recovers many existing methods such as LoRA but also bridges them with the principled algorithmic and theoretical foundations of subspace optimization. This connection highlights a natural "exploration–exploitation" view of subspace methods, guiding the design of new algorithms that achieve strong convergence performance while still preserving memory efficiency. Importantly, our framework establishes the first convergence in the full-parameter space, resolving a critical gap in the current literature where low-rank updates lack such guarantees. We further instantiate the framework into a practical algorithm named PESO-LoRA, based on LoRA-type parameterization. Our algorithm achieves notable improvements over existing methods on standard benchmarks.

1 INTRODUCTION

Pre-training and fine-tuning deep neural networks are the cornerstones of modern AI, powering the success of large-scale foundation models such as Large Language Models (LLMs) (Brown et al., 2020). At their core, both procedures reduce to solving a high-dimensional optimization problem over weight matrices:

$$\Delta W^* := \arg\min_{\Delta W} \ \ell\big(W_0 + \Delta W\big), \tag{1}$$

where $\ell(\cdot)$ is the loss function, W_0 is the initialization, and ΔW the increment. In practice, (1) is typically solved by first-order methods such as Adam (Kingma & Ba, 2014) and AdamW (Loshchilov & Hutter, 2017), which are the workhorses of large-scale training. However, these methods require storing additional optimizer states (e.g., momentum and velocity), and for LLMs this overhead places enormous pressure on memory resources, making parameter-efficient strategies appealing.

In the realm of fine-tuning, we often have limited labeled data for a downstream task but still wish to adapt the pretrained weights effectively and efficiently. Therefore, updating the entire parameter set is both memory-intensive. This motivates the study of Parameter-Efficient Fine-Tuning (PEFT) methods (Han et al., 2024; Houlsby et al., 2019; Hu et al., 2022), where optimization is restricted to a smaller set of parameters initialized from pretrained weights. In other words, W_0 denotes weights obtained from a large-scale pretraining phase, and ΔW is not updated freely but instead follows an efficient parameterization that constrains the search space.

A popular PEFT method is low-rank adaptation (LoRA, Hu et al. (2022)), where matrices in ΔW are expressed as the product of two low-rank factors. LoRA has shown strong empirical success, reducing memory costs while achieving competitive downstream performance. However, it suffers from two key limitations: 1) performance often lags behind full-parameter fine-tuning (Figure 1, left: MetaMathQA); 2) theoretical guarantees are limited, with convergence typically shown only for the low-rank factors (Figure 1, middle: a synthetic example illustrating LoRA's potentially unbounded loss gap). To address these issues, many LoRA variants (Hayou et al., 2024; Wang et al., 2024a;b; Zhang et al., 2023; 2025) have been proposed, yet they largely inherit the same shortcomings and leave the following fundamental question open:

Can we design fine-tuning methods that maintain the memory footprint of LoRA while still enjoying the convergence and optimality of full-parameter fine-tuning?

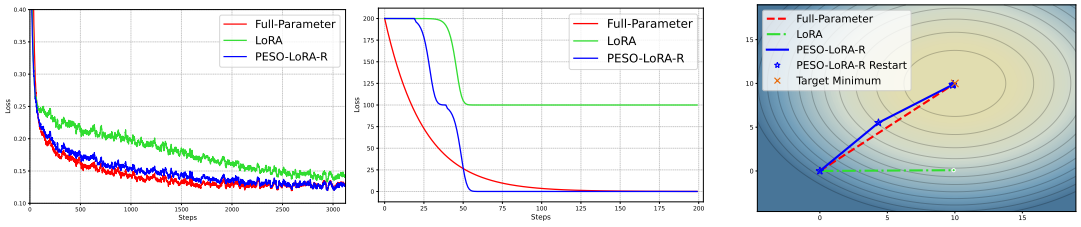

Figure 1: Comparison of full-parameter tuning, LoRA, and our method (PESO-LoRA). Left: Meta-MathQA. Middle: synthetic example $\min_W \|W - M\|_F^2$ with $M = 10 \cdot \mathrm{diag}(1, \ldots, 1, 0, \ldots, 0)$ (r+1 ones); see Appendix A. Right: optimization trajectories. PESO-LoRA bridges the loss gap of LoRA while preserving memory and computation efficiency.

To address this question, we reveal an inherent connection between parameter-efficient training and the classical idea of **subspace minimization**, a long-standing nonlinear optimization strategy dating back to Conn et al. (1994); Cragg & Levy (1969). The central philosophy is to decompose a large-scale problem like (1) into **iterative, simpler** subproblems constrained to carefully chosen subspaces. This view resonates naturally with modern PEFT methods, which restrict updates to structured low-rank forms for better efficiency. Interestingly, subspace minimization historically received less attention in the optimization society, since full-parameter information were often affordable in traditional applications. However, it is especially well suited to LLM training, where *massive dimensionality* calls for *memory-efficient* methods.

Formally, we build on the notion of *intrinsic dimensionality* in LLM training (Aghajanyan et al., 2020; Li et al., 2018), recognized in (Hu et al., 2022) as the origin of LoRA: there exists a dimension-lifting map $\mathcal{M} : \mathbb{R}^d \to \mathbb{R}^{m\times n}$, with $d \ll m \times n$, such that the optimal solution ΔW^* of (1) satisfies

$$\Delta W^* \approx \mathcal{M}(\xi^*), \quad \xi^* := \arg\min_{\xi\in\mathbb{R}^d} \ell(W_0 + \mathcal{M}(\xi)). \tag{2}$$

Here, d stands for the number of trainable parameters, and this characterization implies that it suffices to optimize within the reduced space defined by $\mathcal{M}$ to approximate ΔW^*. For clarity, we focus on a single weight matrix $\Delta W \in \mathbb{R}^{m\times n}$ (multi-layer extensions are straightforward) and represent ξ as a d-dimensional vector. This is without loss of generality, since tensor parameters can always be flattened via vectorization into an isomorphic Euclidean space. For example, LoRA adopts the simple form $\mathcal{M}(A, B) = AB$ with $A \in \mathbb{R}^{m\times r}$, $B \in \mathbb{R}^{r\times n}$, and $d = (m+n)r$. However, it remains unclear whether such a simple $\mathcal{M}$ is sufficient to capture the complexity of LLM training dynamics.

Our framework approximates $\mathcal{M}$ adaptively through a *sequential subspace approximation*, providing a more effective capture of (2). We construct a sequence of maps $\{\mathcal{M}_k\}$, each with a simple representation,

$$\Delta W^* \approx \textstyle\sum_k \mathcal{M}_k(\xi_k^*), \quad \xi_k^* := \arg\min_{\xi\in\mathbb{R}^d} \ell(W_0 + \sum_{i=1}^{k-1} \mathcal{M}_i(\xi_i^*) + \mathcal{M}_k(\xi)). \tag{3}$$

where each $\mathcal{M}_k$ approximates a subspace and ξ_k^* is its low-dimensional coordinate. In essence, the complexity of $\mathcal{M}$ is captured by a sequence of piecewise-linear subspaces. This philosophy parallels classical approximation schemes in numerical analysis such as finite element methods (Bathe, 2006).

Guided by this perspective, we develop a principled framework for PEFT grounded in subspace minimization, named **P**arameter-**E**fficient **S**ubspace **O**ptimization (**PESO**). A key insight is to view the problem (3) through an **exploration–exploitation** lens: *exploration* designs new subspaces that capture full gradient information, while *exploitation* optimizes efficiently within the current subspace. This resolves LoRA's two central limitations: lack of full-parameter convergence guarantees and inefficiency from rigid low-rank parameterization; see Figure 1.

Contributions. Our contributions can be summarized at three levels. Although our focus is on PEFT, many of the ideas developed here naturally extend to pre-training.

I. Perspective Level. We introduce a novel framework PESO for memory-efficient training inspired by classical subspace minimization (Conn et al., 1994), unifying existing PEFT approaches such as LoRA variants (Hu et al., 2022; Wang et al., 2024a;b; Zhang et al., 2023; 2025) and GaLore (Zhao et al., 2024). This framework allows us to explore the rich algorithmic techniques in subspace methods, providing systematic guidance to improve memory-efficient methods. In particular, we highlight two complementary directions: *exploration* of new subspaces through information from the full gradient, and *exploitation* of the current subspace via streaming SVD representations.

II. Theoretical Level. Our exploration mechanism, *full gradient restart*, enables the framework to effectively guide training dynamics. The resulting algorithm is, to our knowledge, the first memory-efficient method for LLM training with provable convergence to full-parameter optimality up to small errors, without additional assumptions such as explicit low-rankness of the solution.

III. Empirical Level. Guided by our PESO framework, we show that two practical instantiations of our framework—`PESO-LoRA-R` and `PESO-LoRA-T`—achieve improved performance while preserving the memory efficiency of state-of-the-art PEFT methods across benchmarks such as GLUE, mathematical reasoning, code generation, and instruction tuning.

Related Work. LoRA (Hu et al., 2022) is perhaps the most widely known PEFT method, and numerous variants have been proposed to enhance its performance. For instance, LoRA+ (Hayou et al., 2024) introduces imbalanced learning rates; PiSSA (Meng et al., 2024) proposes an SVD-based initialization of pretrained weights; and AdaLoRA (Zhang et al., 2023) maintains an adaptive SVD-based low-rank representation. Other extensions focus on gradient scaling (Tastan et al., 2025; Zhang & Pilanci, 2024). More recent work leverages information from the full gradient: LoRA-GA (Wang et al., 2024a) and LoRA-Pro (Wang et al., 2024b) propose memory-efficient gradient approximations, projection-based methods are studied in (Liang et al., 2024; Zhao et al., 2024; Zhu et al., 2024), and LoRA-One (Zhang et al., 2025) employs the SVD of the full gradient for initialization.

Convergence guarantees for PEFT algorithms remain scarce, and existing results typically address only the low-dimensional parameters (Jiang et al., 2024). A related line of work studies *subspace descent* methods (Chen et al., 2025; Kozak et al., 2019; Liang et al., 2024), which constrain updates to $W_{k+1} \leftarrow W_k - \eta_k P_k P_k^\top G_k$, where P_k is the projection matrix, η_k the learning rate, and G_k the full gradient. These approaches establish convergence in the full-parameter space, but under extra structural assumptions. For example, Liang et al. (2024) analyze a continuous-time variant via Lyapunov arguments, but require $P^\top G = 0 \Rightarrow G = 0$, which holds only if P has full column rank—an unrealistic condition when $r < m$. Likewise, Chen et al. (2025); Kozak et al. (2019) rely on random projection theory, assuming $\mathbb{E}[PP^\top] = I_m$ and $P^\top P = I_r$, conditions not needed in our analysis. Closer to LoRA, Jang et al. (2024) provide a convergence analysis, but only within the Neural Tangent Kernel (NTK) regime, limiting its applicability.

Subspace minimization is a classical theme in nonlinear optimization (Conn et al., 1994; Cragg & Levy, 1969; Yuan, 2014). It was historically overshadowed by full-parameter algorithms such as L-BFGS (Liu & Nocedal, 1989) and conjugate gradient methods (Nocedal & Wright, 2006, Ch. 5), since many traditional applications could afford storing full gradients and quasi-Newton pairs. More recently, however, subspace-based strategies have re-emerged in large-scale derivative-free optimization (Cartis & Roberts, 2023; Dzahini & Wild, 2024; Menickelly, 2024; Nozawa et al., 2025; Zhang, 2025), where gradients are unavailable and low-dimensional surrogates are crucial.

2 PESO: A Framework From Subspace Minimization

In this section, we provide a novel perspective of PEFT methods with insights from subspace minimization. We summarize an algorithmic framework **P**arameter-**E**fficient **S**ubspace **O**ptimization (**PESO**) in Algorithm 1, and discuss how it unifies many benchmarks such as LoRA and GaLore.

To build an iterative scheme, a central question is how to represent the weight W at each iteration using low-dimensional representation ξ. In (3), the optimization is expressed through evolving subspaces. At iteration k, we define the *anchored state* $\widetilde{W}_k := W_0 + \sum_{i=1}^{k-1} \mathcal{M}_i(\xi_i^*)$ to encode historical progress, and represent

$$W_k = \widetilde{W}_k + \mathcal{M}_k(\xi_k). \tag{4}$$

Following the design of subspace minimization, PESO considers each $\mathcal{M}_k$ to admit a simple image in the form of a subspace: $\mathcal{S}_k := \{\mathcal{M}_k(\xi) : \xi \in \mathbb{R}^d\}$.

Under representation (4), the evolution of W_k can be viewed as three complementary operations: 1) **exploration**: updating $\mathcal{M}_k$ to select a new subspace $\mathcal{S}_k$, (line 5 of Algorithm 1) 2) **exploitation**: optimizing ξ_k within the current subspace (line 7-8 of Algorithm 1), and 3) updating $\widetilde{W}_k$ to absorb past progress into the anchored weights. These operations mirror the classical paradigm of **subspace minimization** (Conn et al., 1994), where a large-scale problem is solved by iteratively: (i) con-

Algorithm 1 PESO: Generic Framework of **P**arameter-**E**fficient **S**ubspace **O**ptimization

Require: Initialization $W_0 \in \mathbb{R}^{m\times n}$, $\xi_0 \in \mathbb{R}^d$, and $\mathcal{M}_0$; an algorithmic subroutine `UpdateSubspace`, an optimizer `Opt`, frequency K.

Set $k \leftarrow 1$ and $\widetilde{W}_0 \leftarrow W_0$.
while stopping criteria not satisfied **do**
  $(\mathcal{M}_k, \widetilde{W}_k) \leftarrow (\mathcal{M}_{k-1}, \widetilde{W}_{k-1})$.
  if $k-1 \bmod K = 0$ **then** ▷ **Exploration** to new $\mathcal{S}_k$
    $(\mathcal{M}_k, \widetilde{W}_k) \leftarrow \texttt{UpdateSubspace}(\mathcal{M}_{k-1}, \widetilde{W}_{k-1})$.
  end if
  $\Delta\xi_k \leftarrow \texttt{Opt}(\xi_{k-1}, \mathcal{M}_k)$ ▷ **Exploitation** of current $\mathcal{S}_k$
  $\xi_k \leftarrow \xi_{k-1} + \Delta\xi_k$.
  $k \leftarrow k+1$.
end while

structing a subspace based on local information such as gradients, (ii) solving a reduced subproblem within that subspace, and (iii) updating the iterate to incorporate the subspace solution.

In our design, exploration and exploitation directly parallel subspace selection and subproblem optimization, while the anchored state $\widetilde{W}_k$ retains progress from earlier subspaces. In LoRA, $\widetilde{W}$ is fixed at W_0, confining progress to the active subspace. In contrast, updating $\widetilde{W}$ absorbs accumulated contributions back into the parameter space, giving rise to two distinct exploration strategies: *warm-start* and *restart*, which we detail below in Section 2.1.

Leveraging this connection to subspace minimization, we present our generic framework PESO in Algorithm 1. It is important to note that, by selecting corresponding parameterization of $\mathcal{M}_k$, `UpdateSubspace`, and `Opt`, we are able to recover a variety of existing benchmarking methods in parameter-efficient training; see representatives in Table 1. We also remark that Algorithm 1 is equivalent to the classical two-loop subspace minimization scheme (Conn et al., 1994), which we defer to Appendix B in Algorithm 4.

2.1 Subspace Exploration-Exploitation in PESO

Now let us discuss two main components of our framework, subspace exploration and exploitation.

Subspace Exploration. Exploring new subspaces is essential for navigating the full-parameter space under memory restriction. Algorithm 1 carries out exploration by `UpdateSubspace`, which updates both $\mathcal{M}_k$ and $\widetilde{W}_k$. Such updates are often performed lazily every K iterations, as in prior works (Liang et al., 2024; Zhang et al., 2023; Zhao et al., 2024; Zhu et al., 2024).

Depending on how much $\mathcal{M}_k$ is changed, two philosophies arise for how exploration interacts with the low-dimensional ξ_k: *warm-start* and *restart*. These are simply two modes of `UpdateSubspace`:

- **Warm-start.** Preserve ξ_k and keep $\widetilde{W}_k$ fixed. Formally,
$$W_{k+1} = \widetilde{W}_k + \mathcal{M}_{k+1}(\xi_k + \Delta\xi_k). \tag{5}$$
- **Restart.** Absorb the previous contribution into the baseline, $\widetilde{W}_{k+1} \leftarrow \widetilde{W}_k + \mathcal{M}_k(\xi_k)$, and start the new subspace from ξ_{new} (often 0):
$$W_{k+1} = \widetilde{W}_{k+1} + \mathcal{M}_{k+1}(\xi_{\text{new}} + \Delta\xi_k). \tag{6}$$

Intuitively, warm-start provides smoother transitions when consecutive subspaces remain similar, while restart marks a new phase, useful when the optimization geometry changes sharply. In practice, these two modes naturally lead to two main approaches for designing `UpdateSubspace`. A warm-start typically updates the parameterization of $\mathcal{M}_k$ smoothly along an *optimization trajectory*—for example, by applying an Adam step on subspace parameters as in LoRA variants—yielding a gradually evolving subspace. Restart, on the other hand, often *reassigns* $\mathcal{M}_k$ directly using local information such as gradients. This strategy is common in classical optimization; for example, in line search (a one-dimensional subspace method) each iteration resets the step size initialization when a new direction is chosen (Nocedal & Wright, 2006, Ch. 3). It is also used

Table 1: Examples of memory-efficient training methods as instances of PESO.

Methods	ξ	$\mathcal{M}_k(\xi)$	$\mathcal{S}_k$	`UpdateSubspace`	Init.
LoRA	(A, B)	AB	$\{A_kB + AB_k : A \in \mathbb{R}^{m\times r},\ B \in \mathbb{R}^{r\times n}\}$	Adam for A_k, B_k	warm-start
AdaLoRA	Λ	$P_k\Lambda Q_k$	$\{P_k\Lambda Q_k : \Lambda \in \mathbb{R}^{r\times r}$ diagonal$\}$	SGD for P_k, Q_k	warm-start
GaLore	R	P_kR	$\{P_kR : R \in \mathbb{R}^{r\times n}\}$	P_k: left r-SVD of G_k	restart
Kozak et al. (2019)	R	P_kR	$\{P_kR : R \in \mathbb{R}^{r\times n}\}$	randomly sample P_k	restart
Liang et al. (2024)	R	P_kR	$\{P_kR : R \in \mathbb{R}^{r\times n}\}$	online PCA of P_k	warm-start

in LLM training, as in GaLore (Zhao et al., 2024), which periodically resets the subspace via the SVD of the full gradient. Concrete examples of both approaches are summarized in Table 1, and Section 3.1 introduces a new restart scheme leveraging full gradients.

Subspace Exploitation. Between two updates of `UpdateSubspace`, our framework performs K iterations of `Opt` within the current subspace $\mathcal{S}_k$. This amounts to solving the subproblem

$$\min_{\xi\in\mathbb{R}^d} \ \ell(\widetilde{W}_k + \mathcal{M}_k(\xi)) \tag{7}$$

approximately for K steps. In practice, `Opt` is often chosen as Adam.

The philosophy relies on a common belief in classical optimization: during an optimization procedure, once an effective subspace is identified, repeatedly exploiting it for multiple iterations improves efficiency. This principle underlies many classical optimization methods, such as trust-region methods (Nocedal & Wright, 2006, Ch. 4) and L-BFGS-B (Byrd et al., 1995).

2.2 Connection to Existing Benchmarks

While (4) may strike to be abstract, many existing benchmarks for LLM training can naturally fit in it by considering specific subspaces. Here we summarize several notable methods in Table 1.

• *Projected subspace.* A simple way to define a memory-efficient subspace is through low-rank projection, where $\mathcal{M}_k : R \in \mathbb{R}^{r\times n} \mapsto P_kR$ is parameterized by a left-projection matrix $P_k \in \mathbb{R}^{m\times r}$. This formulation can be extended to right-sided or two-sided projections. By applying the chain rule to $\nabla_R \ell(\widetilde{W}_k + P_kR)$, one obtains the projected subspace schemes analyzed in (Kozak et al., 2019; Liang et al., 2024; Zhao et al., 2024; Zhu et al., 2024); see Appendix C for details. Within PESO, GaLore (Zhao et al., 2024), APOLLO (Zhu et al., 2024), and stochastic subspace descent (Kozak et al., 2019) correspond to a *restart* strategy by reassigning P_k, while online subspace descent (Liang et al., 2024) adopts a *warm-start* update of P_k via online PCA.

• *Low-rank subspace.* The LoRA family defines the subspace $\{A_kB + AB_k : A \in \mathbb{R}^{m\times r}, B \in \mathbb{R}^{r\times n}\}$, where the adapters (A, B) jointly serve as both ξ and the parameterization of $\mathcal{M}_k$. Consequently, a single Adam update of (A, B) simultaneously updates the subspace and its coordinates, effectively realizing a *warm-start* scheme with $K = 1$. Many LoRA variants can be viewed as modifications of this generic template: LoRA-Pro (Wang et al., 2024b) applies a different preconditioner, and PiSSA (Meng et al., 2024), LoRA-GA (Wang et al., 2024a), and LoRA-One (Zhang et al., 2025) adjust initialization strategies, while other works modify learning rates or scaling rules. Our framework unifies these designs by interpreting them as *specific choices of* `Opt` *or initialization* within the same subspace structure.

• *SVD subspace.* A principled way to extract low-dimensional structure from matrices (such as ΔW) is through Singular Value Decomposition (SVD), leading to the representation $\mathcal{M}_k : \lambda \in \mathbb{R}^r \mapsto U\mathrm{diag}(\lambda)V$. Here, (U, V) define the subspace (exploration), while λ is the low-dimensional coordinates (exploitation). This separation fits directly into Algorithm 1, enabling flexible optimization strategies for (U, V) and λ. AdaLoRA (Zhang et al., 2023) exemplifies this parameterization, and our framework clarifies the roles of (P_k, Λ_k, Q_k) in their notation. We build on this principle in Section 3, where we propose a practical SVD-based variant `PESO-LoRA-T`.

Together, these three subspace categories illustrate how PESO unifies existing PEFT methods under a single framework, setting the stage for designing new practical algorithms and proving convergence in the following sections.

3 PESO-LoRA: Practical Algorithms from the Framework

The previous section establishes a conceptual link between PEFT methods and classical subspace minimization, providing a unifying interpretation. Building on this perspective, we now develop a concrete algorithm **PESO-LoRA**, which extends LoRA using guidance from our PESO framework. We present two variants: `PESO-LoRA-R` leverages a full gradient restart strategy to improve *exploration* of subspaces, and `PESO-LoRA-T` is a SVD-based method that enhances *exploitation* through more effective optimization within each subspace.

3.1 Full Gradient Restart

We now introduce an important variant of the `UpdateSubspace` subroutine in the restart category (see (6)) that enables convergence to stationarity in the full-parameter space. We design `UpdateSubspace` so that each new subspace $\mathcal{S}_k$ induced by $\mathcal{M}_k$ remains well aligned with the current full gradient G_k. We call this scheme *full gradient restart*:

Full Gradient Restart. Given learning rates $\{\eta_k\}$, whenever $k-1 \bmod K = 0$:

1) Absorb history: $\widetilde{W}_k \leftarrow \widetilde{W}_{k-1} + \mathcal{M}_{k-1}(\xi_{k-1})$.
2) Compute the (stochastic) full gradient $G_k = \nabla_W \ell(\widetilde{W}_k)$.
3) Choose a low rank subspace $\mathcal{S}_k^{\text{FG}}$ depending on G_k.
4) Restart with $\xi_k \leftarrow \xi_k^{\text{new}}$ such that $\mathcal{M}_k(\xi_k^{\text{new}}) = -\eta_k P_{\mathcal{S}_k^{\text{FG}}}(G_k)$.

Here, $P_{\mathcal{S}_k^{\text{FG}}}(G_k)$ denotes the projection of G_k onto $\mathcal{S}_k^{\text{FG}}$. This procedure effectively redefines $\mathcal{M}_k$ so that the new adapter is initialized by a *projected gradient step*:

$$W_k \leftarrow W_{k-1} - \eta_k P_{\mathcal{S}_k^{\text{FG}}}(G_k). \tag{8}$$

Thus, each restart ensures that $\mathcal{S}_k$ captures information from full gradients, with initial progress comparable to a standard SGD step. In the literature on subspace methods, incorporating the full gradient into $\{\mathcal{S}_k\}$ is critical for convergence guarantees (Conn et al., 1994; Zhang, 2025). In particular, one can show that $\|\nabla_W \ell\| \to 0$ provided that $G_k := \nabla_W \ell(W_k)$ lies in $\mathcal{S}_k$. Building on this, we demonstrate in Section 5 that full gradient restart ensures convergence to a stationary point of the original problem (1) by interleaving projected steepest-descent steps with subspace updates.

Algorithm 2 `PESO-LoRA-R`: **PESO** with **LoRA** and Subspace Explo**R**ation

Require: Pre-trained parameters $W_0 \in \mathbb{R}^{m\times n}$, frequency K, scale parameter γ.

Set $k \leftarrow 1$, $\widetilde{W}_0 \leftarrow W_0$, $A_0 \leftarrow 0$ and $B_0 \leftarrow 0$.
while stopping criteria not satisfied **do**
  if $k-1 \bmod K = 0$ **then**
    $\widetilde{W}_k \leftarrow \widetilde{W}_{k-1} + A_{k-1}B_{k-1}$.
    Compute stochastic full gradient G_k.
    $(U_k, \Lambda_k, V_k) \leftarrow \texttt{SVD}(-G_k)$. ▷ Top-$r$ SVD of G_k
    Set $A_{k-1} \leftarrow \frac{1}{\sqrt{\gamma}} U_k \sqrt{\Lambda_k}$ and $B_{k-1} \leftarrow \frac{1}{\sqrt{\gamma}} \sqrt{\Lambda_k} V_k$.
  end if
  $(A_k, B_k) \leftarrow \texttt{AdamW}(A_{k-1}, B_{k-1})$. ▷ One AdamW step on (A_{k-1}, B_{k-1})
  $k \leftarrow k+1$.
end while
return $\widetilde{W}_k + A_k B_k$.

A practical construction of $\mathcal{S}_k^{\text{FG}}$ is to compute a rank-r SVD of G_k and define the subspace as the span of its top singular directions. This ensures that $\mathcal{S}_k^{\text{FG}}$ captures the main structure of G_k, while the approximation error $\|G_k - P_{\mathcal{S}_k^{\text{FG}}}(G_k)\|$ is governed by the spectral tail of G_k. Crucially, this tail is *independent* of the rank gap in the objective, underscoring a key distinction between representation deficiency (e.g., LoRA) and update efficiency. In practice, given (U_k, V_k) from the rank-r SVD of

G_k, one can also restart with $\mathcal{M}_k(\xi_k^{\text{new}}) = -\eta_k U_k V_k$, which corresponds to the update from the recent training benchmark Muon (Jordan et al., 2024), providing improved stability over (8).

One practical advantage of full gradient restart is that it acts as a **"plug-and-play"** mechanism for existing PEFT methods. It can be applied with a moderate frequency K to reduce the cost of SVD while still guiding subspace exploration effectively. Recent work, such as GaLore (Zhao et al., 2024) and LoRA-One (Zhang et al., 2025), has demonstrated the empirical benefits of leveraging the full gradient. In particular, the recent variants LoRA-GA (Wang et al., 2024a) and LoRA-One (Zhang et al., 2025) can be interpreted as special cases of LoRA with full gradient restart applied *only at initialization*. To achieve convergence in the full-parameter space, we propose `PESO-LoRA-R` (Algorithm 2), which embeds LoRA with a repeated restart mechanism every K iterations.

To implement Algorithm 2, directly assigning $(A_{k-1}, B_{k-1}) \leftarrow 1/\sqrt{\gamma}(U_k\sqrt{\Lambda_k}, \sqrt{\Lambda_k}V_k)$ can cause instability due to mismatches in optimization states. For robustness, we propose alignment techniques to maintain consistency of subspace bases, momentum, and velocity; details are provided in Appendix D. Finally, we remark both empirical evidence and theoretical results suggest that gradients G_k in deep learning often have strong low-rank structure, making them especially suitable for efficient SVD-based approximations (Cosson et al., 2023; Yang et al., 2023; Zhao et al., 2024).

3.2 Exploitation via SVD Subspace

Having discussed exploration techniques inspired by subspace minimization, we now turn to the complementary philosophy: exploitation within the current subspace.

As outlined in Section 2.2, an SVD-based parameterization provides a clean and principled way to define $\mathcal{M}_k$. Specifically, we approximate the target mapping $\mathcal{M}(\xi^*)$ by a sum of rank-r components, $\sum_k U_k \xi_k^* V_k$, where each pair (U_k, V_k) defines an *SVD subspace* of rank r. Because SVD naturally captures the dominant gradient directions, this parameterization ensures that exploitation is focused on the most informative directions in the weight space.

Within each subspace, we optimize the low-dimensional coordinate ξ for K steps using Adam. This design can be viewed as an extension of LoRA, with the key difference that the SVD structure explicitly decouples subspace exploitation (through ξ) from exploration (through (U, V)). The practical variant is summarized in Algorithm 3. A small frequency K (e.g., 1 or 2) often suffices for strong performance without significant overhead.

Algorithm 3 `PESO-LoRA-T`: **PESO** with **LoRA** and Subspace Exploi**T**ation

Require: Pretrained weights $W_0 \in \mathbb{R}^{m\times n}$, initial subspace matrices $U_0 \in \mathbb{R}^{m\times r}$, $V_0 \in \mathbb{R}^{r\times n}$, initial coordinate $\xi_0 \in \mathbb{R}^r$, frequency K.

Set $k \leftarrow 1$.
while stopping criterion not met **do**
  Keep $(U_k, V_k) \leftarrow (U_{k-1}, V_{k-1})$.
  if $k - 1 \bmod K = 0$ **then**
    $(U_k, V_k) \leftarrow \texttt{AdamW}(U_{k-1}, V_{k-1})$. ▷ One AdamW step on (U_{k-1}, V_{k-1})
  end if
  $\xi_k \leftarrow \texttt{AdamW}(\xi_{k-1})$. ▷ One AdamW step on ξ_{k-1}
  $k \leftarrow k + 1$.
end while
return $W_0 + U_k \operatorname{diag}(\xi_k) V_k$.

4 Experiments

In this section, we conduct experiments to evaluate our methods across diverse tasks and models, comparing with standard LoRA-based approaches and full fine-tuning. We first assess natural language understanding on the GLUE benchmark (Wang et al., 2018) by fine-tuning T5-base (Raffel et al., 2020). We then evaluate natural language generation on Llama-2-7B (Touvron et al., 2023) for tasks including mathematical reasoning, code generation, and general instruction tuning. Finally, we demonstrate that `LoRA-PESO-R` remains effective even under strict memory constraints when trained for more epochs. Implementation details are provided in Appendix E.

4.1 Natural Language Understanding Tasks

We fine-tune the T5-base model on a subset of GLUE, including MNLI, SST-2, CoLA, QNLI, and MRPC, and evaluate performance using test accuracy (%). Following the setting in (Zhang et al., 2025), we compare our method against several LoRA variants, including LoRA (Hu et al., 2022), LoRA+ (Hayou et al., 2024), P-LoRA (Zhang & Pilanci, 2024), PiSSA (Meng et al., 2024), LoRA-GA (Wang et al., 2024a), LoRA-Pro (Wang et al., 2024b), and LoRA-One (Zhang et al., 2025). For fairness, hyperparameters are tuned individually for each method.

The results are summarized in Table 2. PESO-LoRA-R and PESO-LoRA-T achieve the best performance on three of the five GLUE tasks (MNLI, SST-2, and QNLI), which are also the *larger datasets*. On the remaining tasks, PESO-LoRA-T ranks second. This demonstrates the overall efficiency and robustness of our approaches, with advantages most evident on larger datasets that demand longer training and stronger exploration–exploitation. Moreover, PESO-LoRA-T generally outperforms PESO-LoRA-R, but at the cost of 1.4× more computation time, whereas PESO-LoRA-R runs at nearly the same speed as standard LoRA. Memory costs are comparable across all methods, so the choice ultimately depends on whether performance or efficiency is prioritized.

Table 2: Performance of fine-tuned T5-base on natural language understanding tasks with rank set to 8. Results are reported as accuracy (%) over 3 runs. **Bold** and underline indicate the highest and second-highest accuracies *excluding* PESO-LoRA-T, which is shaded in gray and omitted from direct comparison due to its longer runtime.

Method	MNLI	SST-2	CoLA	QNLI	MRPC
LoRA	$85.30_{\pm 0.04}$	$94.04_{\pm 0.09}$	$72.84_{\pm 1.25}$	$93.02_{\pm 0.07}$	$68.38_{\pm 0.01}$
LoRA+	$85.81_{\pm 0.09}$	$93.85_{\pm 0.24}$	$77.53_{\pm 0.20}$	$93.14_{\pm 0.03}$	$74.43_{\pm 1.39}$
P-LoRA	$85.28_{\pm 0.15}$	$93.88_{\pm 0.11}$	$79.58_{\pm 0.67}$	$93.00_{\pm 0.07}$	$83.91_{\pm 1.16}$
PiSSA	$85.75_{\pm 0.07}$	$94.07_{\pm 0.06}$	$74.27_{\pm 0.39}$	$93.15_{\pm 0.14}$	$76.31_{\pm 0.51}$
LoRA-GA	$85.70_{\pm 0.09}$	$94.11_{\pm 0.18}$	$80.57_{\pm 0.20}$	$93.18_{\pm 0.06}$	$85.29_{\pm 0.24}$
LoRA-Pro	$\underline{86.03}_{\pm 0.19}$	$94.19_{\pm 0.13}$	$\underline{81.94}_{\pm 0.24}$	$\underline{93.42}_{\pm 0.05}$	$\underline{86.60}_{\pm 0.14}$
LoRA-One	$85.89_{\pm 0.08}$	$\underline{94.53}_{\pm 0.13}$	$\mathbf{82.04}_{\pm 0.22}$	$93.37_{\pm 0.02}$	$\mathbf{87.83}_{\pm 0.37}$
PESO-LoRA-R	$\mathbf{86.08}_{\pm 0.15}$	$\mathbf{94.61}_{\pm 0.09}$	$81.50_{\pm 0.16}$	$\mathbf{93.43}_{\pm 0.06}$	$86.36_{\pm 0.11}$
PESO-LoRA-T	$86.09_{\pm 0.04}$	$94.76_{\pm 0.19}$	$82.01_{\pm 0.30}$	$93.45_{\pm 0.03}$	$87.59_{\pm 0.46}$

4.2 Natural Language Generation Tasks

Following prior work (Wang et al., 2024a; Zhang et al., 2025), we fine-tune the Llama-2-7B model on three datasets and evaluate on the corresponding downstream tasks. For mathematical reasoning, we use a 100k subset of MetaMathQA (Yu et al., 2023) and evaluate on GSM8K (Cobbe et al., 2021). For general instruction tuning, we fine-tune on Alpaca (Taori et al., 2023) and evaluate on MMLU (Hendrycks et al., 2020). For code generation, we use a 100k subset of Code-Feedback (Zheng et al., 2024) and evaluate on HumanEval (Chen et al., 2021), reporting PASS@1. To ensure fairness, all datasets are preprocessed to exclude overlaps with test sets. Results are shown in Table 3. Remarkably, our methods outperform baselines on two of the three tasks—mathematical reasoning and code generation—both involving *larger training datasets*, highlighting the substantial gains enabled by subspace exploration and exploitation in handling complex tasks.

4.3 Multi-Epoch Low-Rank Analysis

To demonstrate the effectiveness of subspace exploration, we extend T5-base fine-tuning on SST-2 from one epoch (Section 4.1) to four. This longer schedule enables more thorough exploration and reduces the intrinsic low-rank bottleneck. As shown in Table 4, PESO-LoRA-R with $r = 2$

Table 3: Performance of fine-tuned Llama-2-7B on natural language generation tasks with rank set to 8. Results are reported as accuracy (%) over 3 runs. **Bold** and underline indicate the highest and second-highest accuracies *excluding* PESO-LoRA-T.

	LoRA	LoRA-GA	LoRA-One	PESO-LoRA-R	PESO-LoRA-T
GSM8K	$59.26_{\pm 0.99}$	$56.44_{\pm 1.15}$	$\underline{60.44}_{\pm 0.17}$	$\mathbf{60.55}_{\pm 0.34}$	$60.82_{\pm 0.77}$
MMLU	$45.73_{\pm 0.30}$	$45.15_{\pm 0.57}$	$\mathbf{47.24}_{\pm 0.20}$	$\underline{46.16}_{\pm 0.58}$	$46.44_{\pm 0.37}$
HumanEval	$25.85_{\pm 1.75}$	$26.95_{\pm 1.30}$	$\underline{28.66}_{\pm 0.39}$	$\mathbf{31.70}_{\pm 1.30}$	$30.85_{\pm 1.18}$

Table 4: Performance of fine-tuned T5-base (4 epochs) on the SST-2 dataset. Results are reported as accuracy (%) over 3 runs. **Bold** and underline indicate the highest and second-highest accuracies.

Method	Epoch 1	Epoch 2	Epoch 3	Epoch 4
LoRA ($r = 2$)	$93.02_{\pm 0.44}$	$93.47_{\pm 0.51}$	$93.41_{\pm 0.12}$	$93.52_{\pm 0.17}$
LoRA ($r = 4$)	$94.23_{\pm 0.30}$	$94.42_{\pm 0.15}$	$94.46_{\pm 0.10}$	$94.61_{\pm 0.25}$
LoRA ($r = 8$)	$93.85_{\pm 0.30}$	$94.03_{\pm 0.09}$	$94.54_{\pm 0.05}$	$94.54_{\pm 0.23}$
PESO-LoRA-R ($r = 2$)	$\underline{94.30}_{\pm 0.15}$	$\underline{94.47}_{\pm 0.08}$	$\underline{94.84}_{\pm 0.25}$	$\mathbf{95.14}_{\pm 0.15}$
Full fine-tuning	$\mathbf{94.42}_{\pm 0.11}$	$\mathbf{94.70}_{\pm 0.10}$	$\mathbf{94.85}_{\pm 0.11}$	$\underline{94.90}_{\pm 0.06}$

consistently outperforms standard LoRA even at higher ranks ($r = 4, 8$), showing that it overcomes the low-rank limitation, achieves full-parameter optimality, and delivers stronger performance even under highly restricted memory budgets.

5 Convergence Analysis

In this section, we establish convergence of PESO with the full gradient restart. For general nonconvex losses, the optimality measure $\mathbb{E}\|G_k\|$ converges to zero up to controlled inexactness. To our knowledge, this is the first convergence result for memory-efficient training that guarantees full-parameter stationarity under standard assumptions. We begin by stating the regularity assumptions.

Assumption 1. The loss ℓ is nonconvex, bounded from below, and has L-Lipschitz gradients.

Assumption 2. Stochastic gradients $\widetilde{G}_k$ of the full gradient G_k satisfy $\mathbb{E}(\widetilde{G}_k) = G_k$ and there exists $C > 0$ such that $\mathbb{V}(\widetilde{G}_k) \leq C$.

Assumption 3. The learning rates for full gradient restart in (8) satisfies $\sum \eta_k = \infty$ and $\sum \eta_k^2 < \infty$.

Assumption 4. There exists a sequence $\{\delta_k\} \geq 0$ such that $dist(G_k, \mathcal{S}_k) \leq \delta_k$ for k where full gradient restart is implemmented. Furthermore, $\lim_{k\to\infty} \delta_k < \infty$.

Assumption 5. `Opt` and `UpdateSubspace` generate the updates satisfying $\mathbb{E}[\ell(W_k)] \leq \mathbb{E}[\ell(\widetilde{W}_k + \mathcal{M}_k(\xi_{k-1}))] + C_k$ and $\mathbb{E}[\ell(\widetilde{W}_k + \mathcal{M}_k(\xi_{k-1}))] \leq \mathbb{E}[\ell(W_{k-1})] + C_k$ where $\sum_k |C_k| < \infty$.

Assumptions 1–3 are standard in stochastic optimization; see, e.g., (Bottou et al., 2018). Assumption 4 requires the subspace at full gradient restart to be sufficiently aligned with G_k, allowing approximation errors, e.g., from low-rank SVD. Assumption 5 is mild and holds, for example, when both `Opt` and `UpdateSubspace` use SGD with diminishing learning rates; see Appendix F.

We now state the main convergence result; the proof and its deterministic counterpart are deferred to Appendix F. Importantly, the result holds for **any choice** of `UpdateSubspace`, whether warm-start or restart, and at **any frequency**. In particular, Algorithm 1 may combine different `UpdateSubspace` strategies at varying frequencies, and the guarantee remains valid.

Theorem 5.1. *Suppose all assumptions hold. With full gradient restart, the iterates $\{W_k\}$ generated by Algorithm 1 satisfy* $\liminf_{k\to\infty} \mathbb{E}[\|G_k\|] \leq \lim_{k\to\infty} \delta_k$.

Remark. If $\mathcal{S}_k^{\text{FG}}$ is chosen such that $G_k \in \mathcal{S}_k^{\text{FG}}$, then $\delta_k = 0$ and the optimality measure converges to zero. In PESO-LoRA-R, $\mathcal{S}_k^{\text{FG}}$ is the top-r SVD subspace of $\widetilde{G}_k$, so δ_k reflects both SVD truncation error and a noise term of order $O(\sqrt{C})$. When G_k is effectively low rank and variance-reduction (e.g., online PCA or EMA) is used, $\lim_{k\to\infty} \delta_k$ can be made small.

6 Conclusions and Limitations

This paper bridges classical methodology from nonlinear optimization with the practical challenge of memory-efficient LLM training. It highlights two key perspectives: 1) practical constraints in LLM training, such as memory limits, can motivate specialized optimization designs; 2) principles from nonlinear optimization can in turn guide the development of practical algorithms for LLMs. We believe this opens promising directions for principled and scalable LLM training, while underscoring a broader philosophy: the rapid progress in LLMs can be enriched by classical foundations in computation and optimization. Our study has certain limitations. Due to limited resources, our experiments are restricted to medium-scale settings and do not yet reach the largest practical regimes. Extending our framework to full-scale pre-training remains an important future work, and we expect the methodology developed here to provide a solid foundation for such efforts.

ETHICS STATEMENT

Our research fully adheres to the ICLR Code of Ethics. We did not use any human or animal subjects. All datasets and models were acquired and used in accordance with their respective usage guidelines, and no private data was compromised. This work is free from bias and discriminatory outcomes, avoids using personally identifiable information, and presents no risks to privacy or security. We are committed to conducting this research with complete transparency and integrity.

REPRODUCIBILITY STATEMENT

We take reproducibility seriously and are willing to provide necessary materials to support it. All theoretical results are presented with explicit assumptions, and full proofs are provided in Appendix F. Additionally, experimental settings and implementation details are documented in Appendix A, D and E. Together, these resources allow our claims and results to be verified and reproduced.

REFERENCES

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

Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

Coralia Cartis and Lindon Roberts. Scalable subspace methods for derivative-free nonlinear least-squares optimization. *Mathematical Programming*, 199(1):461–524, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Yiming Chen, Yuan Zhang, Yin Liu, Kun Yuan, and Zaiwen Wen. A memory efficient randomized subspace optimization method for training large language models. *arXiv preprint arXiv:2502.07222*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

AR Conn, Ph L Toint, A Sartenaer, and NIM Gould. On iterated-subspace minimization methods for nonlinear optimization. Technical report, P00024646, 1994.

Romain Cosson, Ali Jadbabaie, Anuran Makur, Amirhossein Reisizadeh, and Devavrat Shah. Low-rank gradient descent. *IEEE Open Journal of Control Systems*, 2:380–395, 2023.

EE Cragg and AV Levy. Study on a supermemory gradient method for the minimization of functions. *Journal of Optimization Theory and Applications*, 4(3):191–205, 1969.

Kwassi Joseph Dzahini and Stefan M Wild. Stochastic trust-region algorithm in random subspaces with convergence and expected complexity analyses. *SIAM Journal on Optimization*, 34(3):2671–2699, 2024.

Athanasios Glentis, Jiaxiang Li, Qiulin Shang, Andi Han, Ioannis Tsaknakis, Quan Wei, and Mingyi Hong. Scalable parameter and memory efficient pretraining for llm: Recent algorithmic advances and benchmarking. *arXiv preprint arXiv:2505.22922*, 2025.

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

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

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, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Uijeong Jang, Jason D Lee, and Ernest K Ryu. Lora training in the ntk regime has no spurious local minima. *arXiv preprint arXiv:2402.11867*, 2024.

Zhanhong Jiang, Nastaran Saadati, Aditya Balu, Minh Pham, Joshua Russell Waite, Nasla Saleem, Chinmay Hegde, and Soumik Sarkar. A unified convergence theory for large language model efficient fine-tuning. In *OPT 2024: Optimization for Machine Learning*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

David Kozak, Stephen Becker, Alireza Doostan, and Luis Tenorio. Stochastic subspace descent. *arXiv preprint arXiv:1904.01145*, 2019.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

Kaizhao Liang, Bo Liu, Lizhang Chen, and Qiang Liu. Memory-efficient llm training with online subspace descent. *arXiv preprint arXiv:2408.12857*, 2024.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

Matt Menickelly. Augmenting subspace optimization methods with linear bandits. *arXiv preprint arXiv:2412.14278*, 2024.

Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

Jorge. Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer, New York, 2nd ed. edition, 2006. ISBN 0387303030.

Ryota Nozawa, Pierre-Louis Poirion, and Akiko Takeda. Zeroth-order random subspace algorithm for non-smooth convex optimization. *Journal of Optimization Theory and Applications*, 204(3): 53, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

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. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Nurbek Tastan, Stefanos Laskaridis, Martin Takac, Karthik Nandakumar, and Samuel Horvath. Loft: Low-rank adaptation that behaves like full fine-tuning. *arXiv preprint arXiv:2505.21289*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024a.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters properly optimized? *arXiv preprint arXiv:2407.18242*, 2024b.

Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Ya-xiang Yuan. A review on subspace methods for nonlinear optimization. In *Proceedings of the International Congress of Mathematics*, pp. 807–827, 2014.

Fangzhao Zhang and Mert Pilanci. Riemannian preconditioned lora for fine-tuning foundation models. *arXiv preprint arXiv:2402.02347*, 2024.

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

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*, 2025.

Zaikun Zhang. Scalable derivative-free optimization algorithms with low-dimensional subspace techniques. *arXiv preprint arXiv:2501.04536*, 2025.

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

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*, 2024.

Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z Pan, Zhangyang Wang, and Jinwon Lee. Apollo: Sgd-like memory, adamw-level performance. *arXiv preprint arXiv:2412.05270*, 2024.

Algorithm 4 Classical Iterated-Subspace Minimization

Require: Initialization $W_0 \in \mathbb{R}^{m\times n}$, $\xi_0 \in \mathbb{R}^d$, $\mathcal{M}_0$, an algorithmic subroutine `UpdateSubspace`, an optimizer `Opt`, frequency K.

1: Set $k \leftarrow 1$ and $\widetilde{W}_0 \leftarrow W_0$.
2: **while** stopping criteria not satisfied **do**
3: Update the subspace $(\mathcal{M}_k, \widetilde{W}_k) \leftarrow \texttt{UpdateSubspace}(\mathcal{M}_{k-1}, \widetilde{W}_{k-1})$.
4: Approximately solve the subspace minimization by `Opt` using K inner-loop iterations:

$$\xi_k^* \leftarrow \text{approx arg}\min_{\xi} \ \ell(\widetilde{W}_k + \mathcal{M}_k(\xi)) \tag{10}$$

5: $W_k \leftarrow \widetilde{W}_k + \mathcal{M}_k(\xi_k^*)$.
6: $k \leftarrow k+1$.
7: **end while**

A Synthetic Example of LoRA Deficiency

One critical limitation in the literature is the absence of convergence guarantees toward valid optimality conditions of (1). Most existing works establish convergence only with respect to the low-dimensional parameters—such as the factors A and B in LoRA—but do not address convergence with respect to the full-parameter matrix W. For instance, Jiang et al. (2024) shows that $\nabla_A \ell(W_0 + A_k B_k)$ and $\nabla_B \ell(W_0 + A_k B_k)$ vanish as $k \to \infty$, while leaving the behavior of $\nabla_W \ell$ uncharacterized. This gap is not merely technical: it highlights a fundamental deficiency of PEFT methods compared to standard full-parameter training. In fact, the optimal loss attained by LoRA can be arbitrarily worse than the true optimal loss of (1). To illustrate this, consider the following simple synthetic example in matrix optimization:

$$\min_{W\in\mathbb{R}^{n\times n}} \quad \|W - M\|_F^2, \quad \text{where } M = a\cdot \text{diag}(1,\dots,1,0,\dots,0), \quad (r{+}1 \text{ ones}). \tag{9}$$

The optimal solution is clearly $W^* = M$ with $f(W^*) = 0$. However, applying LoRA with rank r to (9) can at most achieve a rank-r approximation of M, and attains $f(A^*B^*) = a^2$. As a increases, or as the rank mismatch between the LoRA adapters and the true solution grows, the optimality gap between LoRA and full gradient methods can become arbitrarily large.

This example underscores the cost of memory restrictions: while low-rank parameterizations save memory, they may fundamentally limit convergence to the true optimum. In the middle and right panels of Figure 1, we show that LoRA with exploration (`PESO-LoRA-R`) can effectively converge to the true optimal solution while LoRA would not. Note in this implementation of `PESO-LoRA-R` (Algorithm 2), the SVD of the full gradient is a rank-r SVD and therefore the low-rankness of this computational scheme would not affect the convergence.

B Review on Subspace Minimization

It is worth noting that Algorithm 1 is essentially equivalent to the classical two-loop subspace minimization scheme (Conn et al., 1994), summarized in Algorithm 4.

The key distinction between warm-start and restart, discussed in (5) and (6) of Section 2.1, lies in how the subproblem (10) is initialized within each outer iteration of Algorithm 4.

C Projected Subspace and Memory Efficiency

One-sided projected subspaces in PESO can offer stronger memory efficiency than LoRA. This idea is exemplified by GaLore (Zhao et al., 2024), which we now place in the PESO framework. GaLore requires memory of order $mn{+}mr{+}2nr$ (assuming $m \le n$), compared to LoRA's $mn{+}3mr{+}3nr$.

With the projected subspace representation in Table 1, optimization reduces to $\xi := R \in \mathbb{R}^{r\times n}$. By the chain rule,

$$\nabla_R \ell(W_k) = \nabla_\xi \ell(\widetilde{W}_k + P_k R_k) = P_k^\top \nabla_W \ell(W_k) = P_k^\top G_k. \tag{11}$$

Once P_k is computed and stored, subspace gradients are obtained directly from G_k with no extra overhead, though computing the full G_k each iteration is more costly than subspace-only gradients.

Then, an *exploitation* step in the subspace by gradient descent with learning rate η_k gives

$$\begin{aligned} W_{k+1} &= \widetilde{W}_k + \mathcal{M}_k(R_k - \eta_k P_k^\top G_k) \\ &= \widetilde{W}_k + P_k(R_k - \eta_k P_k^\top G_k) \\ &= W_k - \eta_k P_k P_k^\top G_k, \end{aligned} \tag{12}$$

which matches the classical projected subspace descent step (Kozak et al., 2019).

In GaLore, P_k is chosen as the rank-r left SVD of the full gradient at fixed intervals. PESO recovers GaLore when the subspace gradient $\nabla_R \ell(W_k) = P_k^\top G_k$ in (12) is replaced with its Adam update.

This shows how GaLore saves memory: gradients of ξ_k are derived directly from G_k, and updates are written back into W via (12), reusing the stored pretrained weights W_0. Thus explicit storage of ξ_k is unnecessary. Further savings arise because GaLore omits optimizer states for $\mathcal{M}_k$ (i.e., for P_k), instead updating P_k by direct reassignment in a restart manner. However, smoother transitions of subspace parameters often yield greater stability, as observed in our experiments; we discuss smoothing techniques for restart strategies in the next section.

D Implementation of PESO-LoRA-R

We discuss several implementation details of `PESO-LoRA-R` that are critical for practical stability and performance.

D.1 Smoothing the Subspaces

A potential issue with restart methods (see (6)) is that they directly reassign the subspace parameterization from new information, which can introduce sharp changes and instability, especially in LLM training where stochastic noise is significant.

To mitigate this, we adopt an Exponential Moving Average (EMA) of old and new subspaces, similar in spirit to how Adam (Kingma & Ba, 2014) stabilizes noisy updates. However, this is nontrivial in `PESO-LoRA-R` (Algorithm 2), since the pre-restart adapters (A_k, B_k)—evolved through Adam dynamics—may differ significantly in scale and coordinates from the restarted pair $(U_k\sqrt{\Lambda_k}, \sqrt{\Lambda_k}V_k)$ obtained from rank-r SVD. A naive EMA would mismatch these terms and discard valuable exploration information.

We resolve this by performing *basis and scaling alignment*. For clarity, we omit the subscript k. Given current adapters (A, B), we first compute thin QR factorizations:

$$A = Q_A R_A, \quad B = R_B Q_B^\top, \quad Q_A^\top Q_A = I_r,\ Q_B^\top Q_B = I_r,$$

to extract bases (Q_A, Q_B) and decouple scaling. Next, let the rank-r SVD of the full gradient be $-G \approx U\,\Sigma\,V^\top$. We align (U, V) to (Q_A, Q_B) by applying SVD to $Q_A^\top U$ and $Q_B^\top V$:

$$Q_A^\top U = P_U \Sigma_U Q_U^\top,\ \ R_U := P_U Q_U^\top, \qquad Q_B^\top V = P_V \Sigma_V Q_V^\top,\ \ R_V := P_V Q_V^\top,$$

yielding aligned bases

$$\widehat{U} := U R_U^\top, \qquad \widehat{V} := V R_V^\top.$$

Here R_U solves

$$\min_R \|UR - Q_A\|_F, \quad RR^\top = I,$$

so $\widehat{U}$ best aligns U with Q_A in Frobenius norm; the same holds for $\widehat{V}$. This produces the best alignment by classical low-rank SVD guarantees.

We then smooth the bases via EMA:

$$U_{\text{ema}} := \tau_1 Q_A + (1-\tau_1)\widehat{U}, \qquad V_{\text{ema}} := \tau_1 Q_B + (1-\tau_1)\widehat{V},$$

for smoothing parameter τ_1. To smooth the scaling, we project the old adapter into the new bases and combine with the gradient:

$$S_{\text{new}} := \tau_2\,[U_{\text{ema}}^\top (AB)\, V_{\text{ema}}] - (1-\tau_2)\,[U_{\text{ema}}^\top G\, V_{\text{ema}}],$$

with parameter τ_2. Since this merges the scaling of A and B, we refactorize S_{new} using polar decomposition (Glentis et al., 2025):

$$S_{\text{new}} = R_L\,\Sigma\,R_R^\top,$$

and define the new adapters via balanced splitting:

$$A_{\text{new}} := U_{\text{ema}}\, R_L\, \Sigma^{1/2}, \qquad B_{\text{new}} := \Sigma^{1/2}\, R_R^\top\, V_{\text{ema}}^\top.$$

Finally, although empirically less effective, we also rotate the momentum vectors according to the new basis. Specifically, we compute

$$T_A := Q_A^\top U_{\text{ema}}, \qquad T_B := Q_B^\top V_{\text{ema}},$$

and update the pre-restart momentum (m_A, m_B) as

$$m_A \leftarrow m_A T_A, \qquad m_B \leftarrow T_B^\top m_B.$$

Training then proceeds with the new adapters $(A_{\text{new}}, B_{\text{new}})$ in PESO-LoRA-R.

D.2 Momentum and Velocity Alignment

Even with basis and scaling alignment from the previous subsection, another stability issue arises: the new adapters $(A_{\text{new}}, B_{\text{new}})$ can induce gradients of very different magnitudes compared to the old (A, B). Since restarts are based on the SVD of the full gradient, the new adapters align with top gradient directions, so the gradients with respect to $(A_{\text{new}}, B_{\text{new}})$ are typically larger. This mismatch can leave the velocity "too cold": historical states (v_A, v_B) may underestimate the new gradient magnitudes, leading to an excessively large normalized step and unstable behavior, often observed as jumps in the loss curve.

To address this, we propose a combined *momentum/velocity scaling* technique with a β_2 *warm-up*. Let (m_A, m_B) and (v_A, v_B) denote the momentum and velocity before restart, and (g_A, g_B) the gradients after restart computed with respect to $(A_{\text{new}}, B_{\text{new}})$. We define scaling factors

$$s_A^{(v)} = \frac{\|g_A\|^2}{\|v_A\|}, \quad s_A^{(m)} = \frac{\|g_A\|}{\|m_A\|}, \quad s_B^{(v)} = \frac{\|g_B\|^2}{\|v_B\|}, \quad s_B^{(m)} = \frac{\|g_B\|}{\|m_B\|},$$

which correct scale mismatches between the old optimization states and the new gradients. Here, $\|\cdot\|$ denotes the RMS norm. Momentum and velocity are then rescaled as

$$v_A \leftarrow s_A^{(v)}\, v_A, \quad m_A \leftarrow s_A^{(m)}\, m_A, \quad v_B \leftarrow s_B^{(v)}\, v_B, \quad m_B \leftarrow s_B^{(m)}\, m_B.$$

This resolves scale mismatches, but an additional adjustment is needed: $\beta_2 = 0.999$ (velocity EMA) adapts much more slowly than $\beta_1 = 0.9$ (momentum EMA). At initialization, bias correction balances these, but after a restart we require extra correction. We therefore decrease β_2 immediately after a restart and gradually warm it back to 0.999 over a window T. If a restart occurs at iteration t_r, then for $t_r \le t \le t_r + T$ we set

$$\beta_2(t) = \beta_{2,\min} + \left(\beta_{2,\text{final}} - \beta_{2,\min}\right) \tfrac{1}{2}\left(1 - \cos\tfrac{\pi(t-t_r)}{T}\right), \quad \beta_{2,\text{final}} = 0.999,$$

and for $t > t_r + T$ we use $\beta_2 = 0.999$ as usual. In our experiments, we set $\beta_{2,\min} = 0.95$ and $T = \lfloor K/3 \rfloor$.

E Experimental Details

All experiments are conducted on NVIDIA RTX A6000 GPUs. For PESO-LoRA-R, to further reduce computational cost, we restrict the exploration frequency to two times per epoch.

E.1 Natural Language Understanding

In Section 4.1, we present the results of our methods and various LoRA-based algorithms on natural language understanding tasks, following the prompt tuning configuration of (Wang et al., 2024a). The general hyperparameter settings are kept consistent across all algorithms which are shown in Table 5. To ensure a fair comparison, we follow (Zhang et al., 2025) and tune the learning rates via grid search over $\{1 \times 10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$. Additionally, following the choices of Zhang et al. (2025), the scale parameters for LoRA-One are set to be {128, 16, 128, 128, 64} for {MNLI, SST-2, CoLA, QNLI, MRPC}.

For PESO-LoRA-R, we set the smoothing parameter $\tau_1 = \tau_2 = 0.9$, with frequency K chosen as $\{2000, 500, 100, 500, 40\}$ for {MNLI, SST-2, CoLA, QNLI, and MRPC} based on empirical observations. When $(k-1) \bmod K = 0$ and $k \neq 0$, we set the scale parameter $\gamma = 1$; when $k = 0$, the scale parameter is set the same as in LoRA-One. To further reduce computational cost, we restrict the restart frequency to times per epoch. For PESO-LoRA-T, we set frequency $K = 1$ for all datasets.

Table 5: Common hyperparameters for LoRA fine-tuning on T5-base model.

Epoch	Optimizer	(β_1, β_2)	ϵ	Batch Size	Weight Decay	LR Scheduler
1	AdamW	(0.9, 0.999)	1×10^{-8}	32	0	cosine
Warm-up Ratio	LoRA Alpha	#Runs	Sequence Length	Adapt Precision	Backbone Precision	Gradient Batch Size
0.03	16	3	128	FP32	FP32	8

E.2 Natural Language Generation

For natural language generation tasks in Section 4.2, we follow the configuration of prompt tuning and strategy of hyperparameter tuning in (Zhang et al., 2025) to ensure fair comparison. We search the best learning rate over $\{5\times10^{-4}, 2\times10^{-4}, 1\times10^{-4}, 5\times10^{-5}, 2\times10^{-5}, 1\times10^{-5}\}$, and the general hyperparameter setting is summarized in Table 6. Additionally, following the choice of Zhang et al. (2025), the scale parameters are set to $\{128, 16, 128\}$ for LoRA-One and $\{64, 64, 64\}$ for LoRA-GA.

Table 6: Common hyperparameters for LoRA fine-tuning on Llama-2-7B model.

Epoch	Optimizer	(β_1, β_2)	ϵ	Batch Size	Weight Decay	LR Scheduler
1	AdamW	(0.9, 0.999)	1×10^{-8}	32	0	cosine
Warm-up Ratio	LoRA Alpha	#Runs	Sequence Length	Adapter Precision	Backbone Precision	Gradient Batch Size
0.03	16	3	1024	FP32	BF16	8

For `PESO-LoRA-R`, we set the smoothing parameter $\tau_1 = \tau_2 = 0.9$, with frequency $K = 500$ for all the experiments. When $(k-1) \bmod K = 0$ and $k \neq 0$, we set the scale parameter $\gamma = 1$; when $k = 0$, the scale parameter is set the same as in LoRA-One. For `PESO-LoRA-T`, we set frequency $K = 1$ for all datasets.

E.3 Multi-Epoch Low-Rank Analysis

In Section 4.3, we fine-tune the T5-base model on SST-2 dataset for 4 epochs. We vary the rank of LoRA in $\{2, 4, 8\}$, keep the rank of `PESO-LoRA-R` as 2, and add full-parameter fine-tuning for comparison. We keep all the other hyperparameter settings the same as in E.1.

F Proofs

In this section, we provide the proofs of the theoretical results stated in Section 5. For completeness, we begin with the deterministic case, i.e., when the gradients G_k accessed by PESO (Algorithm 1) are exact, without stochastic noise. We then prove Theorem 5.1, which considers the stochastic setting where only noisy gradients $\widetilde{G}_k$ are available, satisfying Assumption 2.

F.1 Deterministic Case

We first state the deterministic counterpart of Assumption 5:

Assumption 6. `Opt` and `UpdateSubspace` generate the updates satisfying $\ell(W_k) \leq \ell(\widetilde{W}_k + \mathcal{M}_k(\xi_{k-1}))$ and $\ell(\widetilde{W}_k + \mathcal{M}_k(\xi_{k-1})) \leq \ell(W_{k-1})$ for all $k = 1, 2, \cdots$.

Assumption 6 requires that both `Opt` and `UpdateSubspace` act as descent methods, ensuring that the loss is non-increasing. This is the deterministic analogue of Assumption 5, and it holds, for example, when both are implemented as gradient descent with step size $\alpha \leq 1/L$, where L is the Lipschitz constant from Assumption 1. In particular, one can take `UpdateSubspace` to be a warm-start step that performs gradient descent on the subspace parameters with respect to the original loss function. This is formalized below in a standard result from the optimization literature (see, e.g., Nesterov, 2018).

Proposition F.1. *Let* `Opt` *and* `UpdateSubspace` *be gradient descent schemes on ℓ with constant learning rate $\alpha \leq 1/L$. Then Assumption 6 holds.*

We now present the deterministic convergence result.

Theorem F.2. *Suppose Assumptions 1, 4, and 6 hold. With full gradient restart enabled and learning rate $\eta_k = \frac{1}{L}$, the iterates $\{W_k\}$ generated by Algorithm 1 satisfy* $\liminf_{k\to\infty} \|G_k\| \le \lim_{k\to\infty} \delta_k$.

Proof. We assume the frequency of the full gradient restart is K. By the descent lemma, whenever $k-1 \mod K = 0$ (i.e., when a full gradient restart occurs), define

$$\widehat{W}_k := W_{k-1} - \tfrac{1}{L} P_{\mathcal{S}_k}(G_k). \tag{13}$$

Performing a full gradient restart as `UpdateSubspace`, as discussed in (8), is thus equivalent to moving from W_{k-1} to $\widehat{W}_k$. It follows that

$$\begin{aligned} \ell(\widehat{W}_k) = \ell(W_{k-1} - \tfrac{1}{L} P_{\mathcal{S}_k}(G_k)) &\le \ell(W_{k-1}) + \langle G_k, -\tfrac{1}{L} P_{\mathcal{S}_k}(G_k)\rangle + \tfrac{L}{2}\|\tfrac{1}{L} P_{\mathcal{S}_k}(G_k)\|^2 \\ &= \ell(W_{k-1}) - \tfrac{1}{L}\|P_{\mathcal{S}_k}(G_k)\|^2 + \tfrac{L}{2}\|\tfrac{1}{L} P_{\mathcal{S}_k}(G_k)\|^2 \\ &= \ell(W_{k-1}) - \tfrac{1}{2L}\|P_{\mathcal{S}_k}(G_k)\|^2, \end{aligned} \tag{14}$$

where the second equality holds because projection onto a subspace is orthogonal.

Therefore, by Assumption 6, for iterates $i = k, \dots, k+K-1$ (note that $\ell(W_k) \le \ell(\widehat{W}_k)$ since W_k is obtained by applying `Opt` to $\widehat{W}_k$),

$$\tfrac{1}{2L}\|P_{\mathcal{S}_k}(G_k)\|^2 \le \ell(W_{k-1}) - \ell(\widehat{W}_k) \le \ell(W_{k-1}) - \ell(W_i). \tag{15}$$

Importantly, (15) remains valid regardless of how frequently other types of `UpdateSubspace` are applied between full gradient restarts, since all updates preserve the descent property by Assumption 6. In particular, whenever $\mathcal{M}_k$ is updated without a full gradient restart, we have

$$\begin{aligned} \ell(W_{k-1}) - \ell(W_k) &= \ell(\widetilde{W}_{k-1} + \mathcal{M}_{k-1}(\xi_{k-1})) - \ell(W_k) \\ &\ge \ell(\widetilde{W}_{k-1} + \mathcal{M}_{k-1}(\xi_{k-1})) - \ell(\widetilde{W}_k + \mathcal{M}_k(\xi_{k-1})) \\ &\ge 0. \end{aligned} \tag{16}$$

This ensures the chain of inequalities in (15) continues to hold when updates are performed by `OptM`.

Hence, for any integer $k \in \mathbb{N}$,

$$\tfrac{1}{2L}\|P_{\mathcal{S}_{kK+1}}(G_{kK+1})\|^2 \le \ell(W_{kK}) - \ell(W_{(k+1)K}). \tag{17}$$

Here kK and $(k+1)K$ denote integer products.

Since $\{\ell(W_k)\}$ is bounded below (Assumption 1) and monotonically decreasing (Assumption 6 together with the descent lemma at restart points), it converges by the monotone convergence theorem and is Cauchy. Thus $\ell(W_k) - \ell(W_{k+1}) \to 0$, and

$$\tfrac{1}{2L}\|P_{\mathcal{S}_{kK+1}}(G_{kK+1})\|^2 \to 0. \tag{18}$$

Finally, note that

$$\|G_{kK+1}\| \le \operatorname{dist}(G_{kK+1}, \mathcal{S}_{kK+1}) + \|P_{\mathcal{S}_{kK+1}}(G_{kK+1})\| \le \delta_{kK+1} + \|P_{\mathcal{S}_{kK+1}}(G_{kK+1})\| \to \delta, \tag{19}$$

where $\delta := \lim_{k\to\infty} \delta_k$. Therefore, $\liminf_{k\to\infty} \|G_k\| \le \delta$. □

F.2 Stochastic Case

We begin by verifying the validity of Assumption 5. As an illustrative case, suppose `Opt` and `UpdateSubspace` are implemented by SGD with diminishing step sizes $\{\alpha_k\}$ satisfying $\sum_k \alpha_k < \infty$. Let $\widehat{W}_k$ denote the weight after such an update. By the descent lemma (see also (Bottou et al., 2018, Lemma 4.4)), the expected decrease can be bounded as

$$\mathbb{E}[\ell(\widehat{W}_k)] \;\le\; \mathbb{E}[\ell(W_k)] - \alpha_k\big(1 - \tfrac{L\alpha_k}{2}\big)\mathbb{E}\|G_k\|^2 + \tfrac{L}{2}\alpha_k^2 C. \tag{20}$$

For $\alpha_k \le 1/L$, this simplifies to

$$\mathbb{E}[\ell(\widehat{W}_k)] - \mathbb{E}[\ell(W_k)] \;\le\; -\tfrac{\alpha_k}{2}\,\mathbb{E}\|G_k\|^2 + \tfrac{L}{2}\alpha_k^2 C. \tag{21}$$

Taking positive and negative parts, we obtain

$$\big[\mathbb{E}[\ell(\widehat{W}_k)] - \mathbb{E}[\ell(W_k)]\big]_+ \;\le\; \tfrac{L}{2} C\alpha_k^2, \qquad \big[\mathbb{E}[\ell(W_k)] - \mathbb{E}[\ell(\widehat{W}_k)]\big]_+ \;\le\; \mathbb{E}[\ell(W_k)] - \mathbb{E}[\ell(\widehat{W}_k)]. \tag{22}$$

Summing over all k,

$$\begin{aligned}\sum_{k=0}^{\infty}\left[\mathbb{E}[\ell(\widehat{W}_k)]-\mathbb{E}[\ell(W_k)]\right]_+ &\le \tfrac{L}{2}C\sum_{k=0}^{\infty}\alpha_k^2 < \infty,\\ \sum_{k=0}^{\infty}\left[\mathbb{E}[\ell(W_k)]-\mathbb{E}[\ell(\widehat{W}_k)]\right]_+ &\le \sup_k \mathbb{E}[\ell(W_k)]-\ell(W^*) < \infty.\end{aligned} \tag{23}$$

Defining $C_k := \mathbb{E}[\ell(\widehat{W}_k)]-\mathbb{E}[\ell(W_k)]$, we conclude that $\sum_k |C_k| < \infty$, hence Assumption 5 holds.

We are now ready to present the proof of our main result, Theorem 5.1.

Proof of Theorem 5.1. Because $\mathcal{S}_k$ are subspaces, $P_{\mathcal{S}_k}$ is a linear operator, which allows the exchangability with $\mathbb{E}$. Therefore, one has

$$\begin{aligned}\mathbb{E}(\|P_{\mathcal{S}_k}(\widetilde{G}_k)\|^2) &= \|\mathbb{E}(P_{\mathcal{S}_k}(\widetilde{G}_k))\|^2 + \mathbb{V}(P_{\mathcal{S}_k}(\widetilde{G}_k))\\ &= \|(P_{\mathcal{S}_k}(G_k))\|^2 + \mathbb{E}\left(\|P_{\mathcal{S}_k}(\widetilde{G}_k - G_k)\|^2\right)\\ &\le \|(P_{\mathcal{S}_k}(G_k))\|^2 + \mathbb{E}\left(\|\widetilde{G}_k - G_k\|^2\right)\\ &\le \|(P_{\mathcal{S}_k}(G_k))\|^2 + C.\end{aligned} \tag{24}$$

Similar to the deterministic case, We assume the frequency of the full gradient restart is K, and consider $k-1 \mod K = 0$ (i.e., when a full gradient restart occurs). Again, we define

$$\widehat{W}_k := W_{k-1} - \tfrac{1}{L}P_{\mathcal{S}_k}(G_k). \tag{25}$$

By the property of the full gradient restart, we have

$$\begin{aligned}\mathbb{E}[\ell(\widehat{W}_k)] &\le \mathbb{E}[\ell(W_{k-1})] + \left\langle G_k, \eta_k\mathbb{E}[P_{\mathcal{S}_k}(\widetilde{G}_k)]\right\rangle + \frac{L}{2}\mathbb{E}[\|\eta_k P_{\mathcal{S}_k}(\widetilde{G}_k)\|^2]\\ &\le \mathbb{E}[\ell(W_{k-1})] + \eta_k\|P_{\mathcal{S}_k}(G_k)\|^2 + \frac{L\eta_k^2}{2}(\|(P_{\mathcal{S}_k}(G_k))\|^2 + C)\\ &= \mathbb{E}[\ell(W_{k-1})] - \left(\eta_k - \frac{L\eta_k^2}{2}\right)\|P_{\mathcal{S}_k}(G_k)\|^2 + \frac{L\eta_k^2}{2}C,\end{aligned} \tag{26}$$

where the first inequality follows from Assumption 1, and the second follows from the fact that $\mathcal{S}_k$ is a subspace and (24).

Then by Assumption 5, suppose $k-1 \mod K = 0$, and for iterates $i = k+1, \cdots, k+K-1$,

$$\mathbb{E}[\ell(W_i)] \le \mathbb{E}[\ell(W_{i-1})] + 2C_i, \tag{27}$$

where $2C_i$ comes from bounding the scenario where both `Opt` and `UpdateSubspace` operate at i-th iterate. Then summing up the inequalities (26) and (27) for $i = k+1, \cdots, k+K-1$, and use the fact that $\mathbb{E}[\ell(W_k)] \le \mathbb{E}[\ell(\widehat{W}_k)] + C_k$ since W_k is obtained by applying `Opt` to $\widehat{W}_k$, we have

$$\left(\eta_k - \frac{L\eta_k^2}{2}\right)\mathbb{E}[\|P_{\mathcal{S}_k}(G_k)\|^2] - \frac{L\eta_k^2}{2}C - C_k - 2\sum_{i=k+1}^{k+K-1} C_i \le \mathbb{E}[\ell(W_{k-1}) - \ell(W_{k+K-1})]. \tag{28}$$

By Assumption 3, $\eta_k \to 0$ so without loss of generality, we can assume $\frac{L\eta_k}{2} \le \frac{1}{2}$ for any $k \in \mathbb{N}$. Therefore for all integer $k \in \mathbb{N}$, we have

$$\frac{\eta_{kK+1}}{2}\mathbb{E}[\|P_{\mathcal{S}_{kK+1}}(G_{kK+1})\|^2] - \frac{L\eta_{kK+1}^2}{2}C - C_{kK+1} - 2\sum_{i=kK+2}^{(k+1)K} C_i \le \mathbb{E}[\ell(W_{kK}) - \ell(W_{(k+1)K})]. \tag{29}$$

By Assumption 1, there exists a constant C_ℓ so that $C_\ell \le \ell(W)$ for any W. Summing up for $k \in \{1, \cdots, T\}$ one has

$$\begin{aligned}\sum_{k=1}^{T}\frac{\eta_{kK+1}}{2}\mathbb{E}[\|P_{\mathcal{S}_{kK+1}}(G_{kK+1})\|^2] \le &\frac{LC}{2}\sum_{k=1}^{T}\eta_{kK+1}^2 + \sum_{k=1}^{T}C_{kK+1} + 2\sum_{k=1}^{T}\sum_{i=kK+2}^{(k+1)K} C_i\\ &+ \mathbb{E}[\ell(W_K)] - C_\ell.\end{aligned} \tag{30}$$

Taking $T \to \infty$, and note that $\sum_k \eta_k^2 < \infty$ and $\sum_k |C_k| < \infty$, the first and second series on the right hand side of (30) are obviously bounded. For the third series, note that

$$\sum_{k=1}^{\infty} \sum_{i=kK+2}^{(k+1)K} C_i \leq \sum_{k=1}^{\infty} \sum_{i=kK+2}^{(k+1)K} |C_i| \leq \sum_{i=1}^{\infty} |C_i| < \infty. \tag{31}$$

Finally $\|\mathbb{E}[\ell(W_K)] - C_\ell\|$ is obviously bounded for a fixed K, and therefore, one has

$$\sum_{k=1}^{\infty} \eta_{kK+1} \mathbb{E}[\|P_{\mathcal{S}_{kK+1}}(G_{kK+1})\|^2] < \infty. \tag{32}$$

By $\sum_k^{\infty} \eta_k = \infty$ and a contradiction argument, one has $\liminf_{k\to\infty} \mathbb{E}[\|P_{\mathcal{S}_k}(G_k)\|] = 0$. Since $\|G_k\| \leq dist(G_k, \mathcal{S}_k) + \|P_{\mathcal{S}_k}(G_k)\| \leq \delta_k + \|P_{\mathcal{S}_k}(G_k)\| \to \delta$ where $\delta := \lim_{k\to\infty} \delta_k$, the final result follows. □

THE USE OF LARGE LANGUAGE MODELS (LLMS)

LLMs did not play a significant role in the conception of this work. The methodology, problem formulation, and theoretical contributions are entirely original and developed independently by the authors. We made limited use of general-purpose LLM tools (ChatGPT and Gemini) for writing polish and occasional code debugging support. No part of the research ideation, design, or substantive writing relied on LLMs.