Optimizing Fine-Tuning Efficiency: Gradient Subspace Tracking on Grassmann Manifolds for Large Language Models

Sahar Rajabi **Sirisha Rambhatla**
Critical ML Lab, University of Waterloo
Waterloo, ON, Canada
`{srajabi, srambhatla}@uwaterloo.ca`

Abstract

Training and fine-tuning Large Language Models (LLMs) demand significant computational resources and time due to their large model sizes and optimizer states. To mitigate these challenges and improve accessibility, several memory-efficient methods have been developed. Methods such as Low-Rank Adaptation (LoRA) optimize model weights within a low-rank subspace, while Gradient Low-Rank Projection (GaLore) projects gradients into a lower-dimensional space to decrease memory footprint. In this paper, we propose Gradient Subspace Tracking (SubTrack), a method that confines optimization to a compact core subspace of the gradient matrices and dynamically tracks its changes using the geometry of Grassmannian manifolds. SubTrack efficiently updates its subspace estimation by leveraging estimation errors and previously identified subspaces. Our results demonstrate that even with rank-1 updates to the underlying subspace, SubTrack achieves comparable or superior performance to GaLore, while **reducing runtime by approx. 15% on an average and up to 20.56% on some datasets**.

1 Introduction

Large Language Models (LLMs) have achieved state-of-the-art performance across various tasks and are rapidly growing in popularity. However, their training and fine-tuning demand substantial resources, such as hardware and time, making them impractical for many applications and contributing to a larger carbon footprint [32, 12, 22, 20, 21, 10, 16]. As a result, there is an acute need to develop memory- and time-efficient methods to democratize their use and mitigate environmental impact. Various techniques have been proposed to reduce memory usage, such as gradient checkpointing [5] and memory offloading [23]. In this context, several Parameter-Efficient Fine-Tuning (PEFT) approaches aim to reduce memory usage by optimizing a subset of model parameters or operating within a lower-dimensional space [7, 29, 17, 24, 27, 20, 11]. Notably, the well-known method LoRA [11] decomposes weight matrices into two low-rank trainable matrices, optimizing network parameters within a small subspace, which significantly reduces the memory footprint.

Memory requirements extend beyond trainable parameters, with a significant portion consumed by optimizers for storing element-wise states and parameters [32]. To address this, recent efforts have focused on reducing the memory footprint of optimizer parameters [15, 1, 19, 6, 31, 21, 32, 22]. GaLore [32] reduces the memory usage of optimizers by projecting gradient matrices into a low-rank subspace and tracking changes via periodic Singular Value Decomposition (SVD) to obtain a rank-r approximation. However, this approach faces several challenges. First, SVD is computationally expensive, and if the gradient does not evolve within a nearly constant subspace, GaLore must increase the frequency of SVD operations, significantly increasing the amount of computation. This is problematic because not all layers' gradients converge to a stable subspace early in training [12].

Moreover, applying SVD to a single gradient matrix is susceptible to data noise [26], and GaLore does not leverage 1) information from the orthogonal space as feedback to adjust the subspace [21] or 2) previously computed subspaces to incorporate past knowledge, which could help mitigate these effects and improve convergence speed.

To address these challenges, we propose Gradient Subspace Tracking (SubTrack), a Grassmannian-based subspace tracking method that efficiently updates the subspace using rank-1 updates. SubTrack accumulates gradients between two update steps to reduce noise effects and leverages information from the orthogonal complement to improve subspace estimation through simple linear algebra operations, which are computationally more efficient than GaLore as they avoid periodic SVD on the main gradient matrices. Furthermore, SubTrack dynamically adapts to changes in the gradient subspace, reducing abrupt shifts in subspace updates for faster convergence. Our main contributions are as follows:

- We introduce SubTrack, a computationally and memory-efficient method that projects gradients onto a core subspace and dynamically adjusts this subspace using Grassmannian manifold geometry, leveraging estimation errors as a signal for adjustment.
- We demonstrate that SubTrack achieves performance comparable to or better than GaLore, with a significant reduction in runtime.
- We prove that our method aligns with GaLore's convergence guarantees while enabling more frequent subspace updates by exercising greater control over subspace adjustments through the use of prior knowledge and errors from the orthogonal space.

2 Related Works

Several works aim to improve the efficiency of training and fine-tuning LLMs, addressing a growing demand as their popularity rapidly increases. LoRA [11], a widely recognized method for reducing the number of trainable parameters, projects model weights into a lower-dimensional space, resulting in two trainable low-rank matrices. This approach optimizes the matrices and significantly reduces memory requirements for fine-tuning large models. Dettmers et al. [7] builds on LoRA by employing quantization techniques and paged optimizers to further reduce memory usage. Additionally, Yaras et al. [29] introduces Deep LoRA, which uses deep matrix factorization for low-rank optimization, addressing overfitting issues and reducing the need for precise tuning of the rank parameter. Several other works have also extended LoRA to enhance the efficiency of training and fine-tuning large models [17, 24, 27]. Miles et al. [20] proposes compressing intermediate activation vectors and reconstructing them during backpropagation to enhance memory efficiency. Additionally, Hao et al. [10] demonstrates that full-parameter fine-tuning is feasible by using random projections on the gradient matrix, showing that LoRA essentially performs a down-projection of the gradient.

Several approaches aim to reduce memory consumption in optimizers, as optimizers like Adam [14] account for a significant portion of memory usage due to their storage of element-wise states to improve the optimization process [15, 1, 19, 6, 31]. MicroAdam [21] tackles this issue by compressing the gradient space for optimization and utilizing the resulting compression error through feedback loops to improve the optimization process. Gur-Ari et al. [9] suggests that a substantial portion of gradients lies within a small, largely consistent subspace, a finding also reported by other studies, including Schneider et al. [25], Yaras et al. [28]. GaLore [32] leverages this property to reduce the memory requirements of optimizers by projecting gradients into a lower-dimensional subspace and then projecting them back for complete parameter tuning. This approach has been effectively integrated with other methods to further reduce memory usage during training and fine-tuning of LLMs [16]. However, not all layers' gradients evolve within a stable low-rank subspace. Jaiswal et al. [12] identifies layers with constantly changing gradients where low-rank projection may be inefficient for tuning. By analyzing the distribution of singular values across different layers, they select those that evolve within a small subspace for fine-tuning while freezing the remaining layers. Gradient Structured Sparsification (Grass) [22] further reduces memory usage by applying sparse projection matrices to the gradient, transforming the gradient matrix into a sparse vector space. This approach leverages sparse representations to significantly decrease the memory footprint.

When working with high-dimensional data, a common approach is to project the data into a lower-dimensional space, and many studies focus on tracking these subspaces as they evolve over time. Balzano et al. [2] introduces an incremental method for updating subspaces on the Grassmannian manifold when data is partially observed. Zhang and Balzano [30] and Kasai [13] address the challenges posed by noise in data streams and evolving environments, proposing methods to handle such noise. Additionally, Blocker et al. [4] presents a method for time-varying data based on geodesics in Grassmannian space to track changes and update the subspace effectively.

3 SubTrack: Tracking the Gradient Subspace

Since gradients typically evolve within a small subspace, compressing this space can significantly reduce the memory footprint of the optimizer. As demonstrated in Zhao et al. [32] whenever the gradient takes the general form

$$G = \sum_i A_i + \sum_i B_i W C_i \tag{1}$$

where i denotes the batch index, and B_i and C_i are positive semi-definite (PSD) matrices, this gradient can be projected onto a small subspace that remains nearly stable while ensuring that the optimization process continues to converge, as discussed in Section 4 and Appendix B. However, the gradient's subspace is not always stable, making it crucial to track its changes for effective optimization. GaLore [32] addresses this by periodically performing SVD on gradient matrices, while keeping the update frequency low to align with the assumption of a stable subspace. However, this approach poses several challenges: **1)** not all gradients converge to a stable subspace within a few iterations, **2)** SVD is computationally expensive, and increasing the frequency of updates to capture changes significantly raises both runtime and environmental costs, and **3)** increasing the update frequency contradicts the assumption of a stable subspace, as SVD is sensitive to noise and does not account for the previously computed subspace or estimation error to regulate the extent of change applied.

We propose Gradient Subspace Tracking, or SubTrack, a computationally efficient method for tracking gradient subspaces. SubTrack utilizes both the orthogonal space and the previously computed subspace to update the core subspace. As we restrict subspace updates to rank-1 changes, we can effectively controls the amount of change in the subspace, enabling more frequent updates without compromising the stability assumption. The subspace is initialized using SVD as follows:

$$G_0 = USV^\top \approx \sum_{i=1}^{r} s_i u_i v_i^\top, \quad P_0 = [u_1, u_2, ..., u_r], \quad Q_0 = [v_1, v_2, ..., v_r]. \tag{2}$$

Here, G_0 is an $m \times n$ gradient matrix at step 0, and U, S, and V are its SVD components, with r denoting the rank. At each optimization step, the gradients are projected onto the subspace of the left singular vectors if $m \leq n$, or onto the right singular vectors otherwise, thereby optimizing memory usage [32]. The optimization is performed within this subspace, after which the gradient is projected back to allow full parameter tuning. For simplicity, we assume $m \leq n$ without loss of generality, implying that $S_0 = P_0$, an $m \times r$ orthonormal matrix whose columns span the underlying subspace.

At each iteration of pre-training or fine-tuning, the matrix S_t, representing the subspace at step t, projects the gradient matrix G_t onto itself by $\widetilde{G}_t = S_t^\top G_t$, where $\widetilde{G}_t$ will be a reduced $r \times n$ matrix representing the projection of the original gradient onto a rank-r subspace. The optimizer then performs optimization within this low-rank space, which substantially reduces the number of state parameters and thus the memory usage. The optimizer outputs $\widetilde{G}_t^O$, which is then projected back to the original space by $\widehat{G}_t = S_t \widetilde{G}_t^O$ to be passed to the network.

As previously discussed, the gradient does not always evolve within a stable low-rank subspace; hence, S_t, the orthonormal matrix spanning the core subspace, must be appropriately updated. In SubTrack, we propose updating the subspace by moving along Grassmannian geodesics. This approach leverages the previously computed subspace and the estimation error from earlier steps to minimize abrupt changes and noise effects.

The underlying subspace is updated after a fixed subspace update interval of k steps. Let T_i denote the i-th subspace update step for $i \in \{1, 2, 3, \ldots\}$. To leverage the estimation error in the orthogonal space and reduce the impact of noise, SubTrack computes an accumulated gradient by averaging the gradients between two consecutive subspace update steps, as shown below:

$$G_{acc} = \frac{1}{T_n - T_{n-1}} \sum_{\substack{t= \\ T_{n-1}}}^{T_n} G_t \tag{3}$$

Algorithm 1 SubTrack

```
Require: Sequence of $m \times n$ gradients $G_t$ with $m \leq n$ (w.l.o.g.), step-size $\eta$, rank $r$, subspace
  update steps k
  Initialize Subspace via SVD Decomposition:
  $P_0 \leftarrow U[:, : r]$ , where $U, S, V \leftarrow \text{SVD}(G_0)$
  $S_0 \leftarrow P_0$                                              {The initial subspace}
  $G_{acc} = 0_{m \times n}$                                         {To keep the accumulated gradient}
  for $t = 1, \ldots, T$ do
    if $t \mod k = 0$ then
      Prepare accumulated gradients: $G_{acc} = \frac{G_{acc}+G_t}{k}$
      Update subspace:
      $G_{lr} = \arg\min_A \|(S_{t-1}A - G_{acc})\|^2$                     {Solving the least square problem}
      $R = G_{acc} - S_{t-1}G_{lr}$                                       {Computing the residual}
      $\nabla F = -2RG_{lr}^\top \approx \widehat{U}_F \widehat{\Sigma}_F \widehat{V}_F^\top$      {Computing the rank-1 estimation of tangent vector}
      $S_t = (S_{t-1}\widehat{V}_F \quad \widehat{U}_F) \begin{pmatrix} \cos \widehat{\Sigma}_F \eta \\ \sin \widehat{\Sigma}_F \eta \end{pmatrix} \widehat{V}_F^\top + S_{t-1}(I - \widehat{V}_F \widehat{V}_F^\top)$      {Updating the subspace}
      Reset accumulated gradients: $G_{acc} = 0_{m \times n}$
    else
      Keep using previous subspace: $G_{acc} = G_{acc} + G_t$ , $S_t = S_{t-1}$
    Return final projected gradient to the optimizer: $S_t^\top G_t$
```

We frame the problem of identifying the subspace as selecting the appropriate element from the Grassmannian, the set of all d-dimensional subspaces within an n-dimensional vector space [3]. Our objective is to minimize the Euclidean distance between the current subspace and the accumulated gradient G_{acc} observed at each update step. The cost function is defined as:

$$F(S_t) = \min_A \|S_t A - G_{acc}\|_F^2, \tag{4}$$

where A is the solution to the least squares problem. The derivative of this function with respect to S_t is given in (5), and $R = G_{acc} - S_t A$ lies in the orthogonal complement of S_t. To update the subspace in the appropriate direction, we compute the tangent vector ∇F, as shown in (6) based on Edelman et al. [8], where the second equality holds because R is orthogonal to $S_t S_t^\top$.

$$\frac{\partial F}{\partial S_t} = 2(S_t A - G_{acc})A^\top = -2RA^\top \tag{5}$$

$$\nabla F = (I - S_t S_t^\top)\frac{\partial F}{\partial S_t} = \frac{\partial F}{\partial S_t} = -2RA^\top \approx \widehat{U}_F \widehat{\Sigma}_F \widehat{V}_F^\top \tag{6}$$

The tangent vector ∇F provides the direction for adjusting the subspace by accounting for the error lying in the orthogonal complement. However, to minimize changes to the subspace, SubTrack first computes a rank-1 approximation of ∇F, determined by its largest singular value and the corresponding singular vectors obtained from its SVD, represented as $\widehat{U}_F \widehat{\Sigma}_F \widehat{V}_F^\top$ in the final equality of (6). This approximation is then used to update the subspace.

As shown by Edelman et al. [8], Bendokat et al. [3], we can move along the Grassmannian geodesic in the direction determined by the computed tangent vector, taking a step of size η, as presented in (7).

$$S_{t+1}(\eta) = (S_t \widehat{V}_F \quad \widehat{U}_F) \begin{pmatrix} \cos \widehat{\Sigma}_F \eta \\ \sin \widehat{\Sigma}_F \eta \end{pmatrix} \widehat{V}_F^\top + S_t(I - \widehat{V}_F \widehat{V}_F^\top) \tag{7}$$

This update step preserves the orthonormality of S_{t+1}, ensuring it remains on the Grassmannian manifold. The last term in (6), which is required when using thin-SVD instead of compact-SVD, projects the previous subspace onto the orthogonal complement of $\widehat{V}_F$. This ensures that the portion of the subspace of S_t not updated in this step is still included. By leveraging the geometry of the Grassmannian manifold, SubTrack effectively tracks the underlying subspace of the gradient space. The pseudo-code for this method is provided in Algorithm 1.

4 Theoretical Analysis

In this section, we analyze the convergence of SubTrack using theoretical analysis. To begin, using SubTrack, the update rule for the weights of the networks is as follows:

$$W_t = W_0 + \sum_{t'=0}^{t'=t-1} \widehat{G}_{t'} \tag{8}$$

Using the full projection, $\widehat{G}_{t'}$ will be computed as shown below, where $S^l_{t'}$ and $S^r_{t'}$ are the rank-r left and right projection matrices.

$$\widehat{G}_{t'} = S^l_{t'} \rho_{t'} ({S^l_{t'}}^\top G_{t'} S^r_{t'}) {S^r_{t'}}^\top \tag{9}$$

L-continuity A function $f(X)$ has Lipschitz-continuity (L-continuity) if for any X_1 and X_2, $\|f(X_2) - f(X_1)\|_F \leq L\|X_2 - X_1\|_F$.

Convergence of SubTrack Suppose gradient has the following form (also (1)) with functions A_i, B_i, and C_i being L-continuous with constants L_A, L_B, and L_C w.r.t. weight matrix W_t; and $\|W_t\|_F \leq M$; where W_t denotes the weight matrix at step t, and M is a scalar value,

$$G = \sum_i A_i + \sum_i B_i W C_i.$$

Now, define $\widehat{B}_{i,t} = (S^l_{i,t})^\top B_i(W_t) S^l_{i,t}$ and $\widehat{C}_{i,t} = (S^r_{i,t})^\top C_i(W_t) S^r_{i,t}$, where $S^l_{i,t}$ and $S^r_{i,t}$ are the rank-r left and right projection matrices; $B_i(W_t)$ and $C_i(W_t)$ denote the dependence of B_i and C_i on the weight matrices W_t. Further letting $P_t = {S^l_t}^\top G_t S^r_t$, and $\kappa_t = \frac{1}{N} \sum_i \lambda_{min}(\widehat{B}_{i,t}) \lambda_{min}(\widehat{C}_{i,t})$, where $\lambda_{min}(\cdot)$ denotes the minimum eigenvalue over each batch, and assuming that the projection matrices remain constant during the training. Then for learning-rate μ and $min(\kappa_t) > (L_A + 2L_B L_C M^2)$, the SubTrack, with $\rho_t \equiv 1$ (the element-wise regularizer of the optimizer) satisfies:

$$\|P_t\|_F \leq [1 - \mu(\kappa_{t-1} - L_A - 2L_B L_C M^2)] \|P_{t-1}\|_F.$$

That is, $P_t \to 0$ and SubTrack converges.

The proof of this theorem is provided in Appendix B, based on Zhao et al. [32]. Note that while both GaLore and SubTrack assume that the projection matrices remain unchanged for the proof of convergence, GaLore must limit the number of updates to ensure convergence, as each update can potentially change the subspace entirely. In contrast, SubTrack leverages only rank-1 updates to the subspace, preventing drastic changes with each update. While a deeper analysis of slowly changing subspaces and their impact on convergence remains an open problem, in practice, this allows SubTrack to perform more updates than GaLore.

The Grassmannian update rule presented in (7) is a direct application of Grassmann geometry [8, 3].

Exponential Map $\exp_p : T_pM \to M$ on a Riemannian manifold M is a mapping that assigns to each tangent vector $\Delta \in T_pM$ the point $\gamma(1) \in M$, where T_pM is the tangent space of M at p, and γ is the unique geodesic originating at p with initial velocity Δ. This map establishes a relationship between geodesics and the Riemannian exponential, such that $\gamma(t) = \exp_p(t\Delta)$ for $t \in \mathbb{R}$.

Stiefel Manifold $\text{St}(n, p)$ parametrizes the set of all $n \times p$ matrices U, with orthonormal columns, each representing a rank-p subspace of $\mathbb{R}^n$.

Grassmann Manifold $\text{Gr}(n, p)$ parametrizes the set of all p-dimensional subspaces of $\mathbb{R}^n$. Each point can be represented by a projection matrix $P = UU^T$, where $U \in \text{St}(n, p)$.

Grassmann Exponential Let $P = UU^T \in \text{Gr}(n, p)$ be a point on the Grassmannian manifold, where $U \in \text{St}(n, p)$ is the orthonormal basis of the corresponding subspace. Consider a tangent vector $\Delta \in T_P\text{Gr}(n, p)$, and let Δ_U^{hor} denote the horizontal lift of Δ to the horizontal space at U in the Stiefel manifold $\text{St}(n, p)$. Suppose the thin SVD of Δ_U^{hor} is given by $\Delta_U^{\text{hor}} = \hat{Q}\Sigma V^T$, where $\hat{Q} \in \text{St}(n, r)$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$ contains the nonzero singular values of Δ_U^{hor} with $r = \min(p, n - p)$, and $V \in \text{St}(p, r)$. The Grassmann exponential map, representing the geodesic emanating from P in the direction Δ, is given by:

$$\text{Exp}^{\text{Gr}}_P(t\Delta) = \left[UV\cos(t\Sigma)V^T + \hat{Q}\sin(t\Sigma)V^T + UV_\perp V_\perp^T\right],$$

Table 1: Evaluating SubTrack and GaLore on their performance and runtime when fine-tuning RoBERTa-Base on GLUE tasks for ranks 8. All hyperparameters including scale-factor and subspace update interval are the same. SubTrack achieved better average performance compared to GaLore while spending a considerably less time for fine-tuning. The performance is measured after fine-tuning for 30 epochs. Wall-times are reported in seconds, and are measured after fine-tuning for 2500 iterations, removing the evaluation steps while performing exactly 5 updates for having a fair comparison between these two methods.

		COLA	STS-B	MRPC	RTE	SST-2	MNLI	QNLI	QQP	Avg
GaLore [32]	Perf.	58.54	90.61	91.30	74.37	94.50	87.34	92.71	87.99	84.67
Rank = 8	Time	188.0	195.6	196.5	328.3	187.1	208.0	217.1	189.4	213.7
SubTrack (Ours)	Perf.	58.54	90.87	91.43	76.53	94.27	87.09	92.49	87.57	**84.85**
Rank = 8	Time	149.4	159.1	171.2	304.9	148.9	177.1	192.1	156.8	**182.4**
Reduction in Wall-Time		**20.56%**	**18.68%**	**12.87%**	**7.11%**	**20.37%**	**14.89%**	**11.48%**	**17.21%**	**15.40%**

where $V_\perp \in \mathbb{R}^{p\times(p-r)}$ is any orthogonal complement of V.

The proof of this theorem can be found in Appendix C. Leveraging this theorem, and incorporating our notation, one can easily verify that the subspace update rule is as stated in (7). This update rule generally converges to a stable subspace if the step size η decreases over time [2]. However, a decreasing step size can impair the ability to accurately track and adapt to subspace changes. Consequently, SubTrack uses a constant step size during training and fine-tuning to effectively adjust subspaces. This approach does not hinder convergence, as proved in the theorem, which guarantees convergence as long as changes are controlled to maintain the stable subspace assumption.

5 Experiments

To ensure a fair comparison of computational efficiency between SubTrack and GaLore, we fine-tuned RoBERTa-Base [18] measuring the corresponding wall-time and performance while keeping all shared hyperparameters consistent. Wall-time is the real-world elapsed time it takes for a process or operation to complete, measured from start to finish, including both the actual runtime and any waiting time for resources or data retrieval. To compare wall-time performance, RoBERTa-Base was fine-tuned on GLUE tasks for 2500 iterations, with the subspace update interval set to 500 iterations and rank 8. Consequently, both methods update the underlying subspace exactly five times, and evaluation steps were excluded to ensure an accurate runtime comparison. For performance evaluation, the model was fine-tuned on these tasks for 30 epochs, with results presented in Table 1. As shown, SubTrack reduces runtime by up to 20.56%. Despite restricting updates to rank-1 changes to the previous subspace, Table 1 demonstrates that SubTrack achieves performance comparable to or even surpassing that of GaLore. Further experimental details are provided in Appendix A.

6 Discussion and Conclusion

We proposed a computationally efficient method that projects gradients into a lower-dimensional subspace, updating this subspace by tracking its changes over time. SubTrack preserves the previously computed subspace and incorporates gradient components from the orthogonal complement to perform rank-1 updates. This approach reduces the frequency of abrupt transitions between iterations and leverages the available information effectively.

In some cases, the extent of changes in the subspace may require updates of rank greater than 1. During our experiments, we observed that applying updates as per (7), using the SVD of the tangent vector from (6), can hinder convergence if the singular values of the tangent vector become very small. Furthermore, simultaneously updating subspace dimensions associated with sufficiently large singular values caused convergence issues in some cases. Therefore, we restricted updates to rank-1 in this paper, as this approach still enabled SubTrack to achieve performance comparable to or better than GaLore, with reduced runtime. In future work, we plan to explore increasing the rank of updates without compromising convergence. Additionally, dynamically selecting the step size could eliminate the need for manual tuning as a hyperparameter and further enhance convergence.

References

[1] Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory-efficient adaptive optimization, 2019. URL `https://arxiv.org/abs/1901.11150`.

[2] Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information, 2011. URL `https://arxiv.org/abs/1006.4046`.

[3] Thomas Bendokat, Ralf Zimmermann, and P.-A. Absil. A grassmann manifold handbook: basic geometry and computational aspects. *Advances in Computational Mathematics*, 50(1), January 2024. ISSN 1572-9044. doi: 10.1007/s10444-023-10090-8. URL `http://dx.doi.org/10.1007/s10444-023-10090-8`.

[4] Cameron J. Blocker, Haroon Raja, Jeffrey A. Fessler, and Laura Balzano. Dynamic subspace estimation with grassmannian geodesics, 2023. URL `https://arxiv.org/abs/2303.14851`.

[5] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. URL `https://arxiv.org/abs/1604.06174`.

[6] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization, 2022. URL `https://arxiv.org/abs/2110.02861`.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

[8] Alan Edelman, T. A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints, 1998. URL `https://arxiv.org/abs/physics/9806030`.

[9] Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace, 2018. URL `https://arxiv.org/abs/1812.04754`.

[10] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors, 2024. URL `https://arxiv.org/abs/2402.03293`.

[11] 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.

[12] Ajay Jaiswal, Lu Yin, Zhenyu Zhang, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. From galore to welore: How low-rank weights non-uniformly emerge from low-rank gradients, 2024. URL `https://arxiv.org/abs/2407.11239`.

[13] Hiroyuki Kasai. Fast online low-rank tensor subspace tracking by cp decomposition using recursive least squares from incomplete observations, 2017. URL `https://arxiv.org/abs/1709.10276`.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL `https://arxiv.org/abs/1412.6980`.

[15] Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states, 2023. URL `https://arxiv.org/abs/2309.01507`.

[16] Pengxiang Li, Lu Yin, Xiaowei Gao, and Shiwei Liu. Owlore: Outlier-weighed layerwise sampled low-rank projection for memory-efficient llm fine-tuning, 2024. URL `https://arxiv.org/abs/2405.18380`.

[17] Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Relora: High-rank training through low-rank updates, 2023. URL `https://arxiv.org/abs/2307.05695`.

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL `https://arxiv.org/abs/1907.11692`.

[19] Kai Lv, Yuqing Yang, Tengxiao Liu, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8187–8198, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.acl-long.445`.

[20] Roy Miles, Pradyumna Reddy, Ismail Elezi, and Jiankang Deng. Velora: Memory efficient training using rank-1 sub-token projections, 2024. URL `https://arxiv.org/abs/2405.17991`.

[21] Ionut-Vlad Modoranu, Mher Safaryan, Grigory Malinovsky, Eldar Kurtic, Thomas Robert, Peter Richtarik, and Dan Alistarh. Microadam: Accurate adaptive optimization with low space overhead and provable convergence, 2024. URL `https://arxiv.org/abs/2405.15593`.

[22] Aashiq Muhamed, Oscar Li, David Woodruff, Mona Diab, and Virginia Smith. Grass: Compute efficient low-memory llm training with structured sparse gradients, 2024. URL `https://arxiv.org/abs/2406.17660`.

[23] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020. URL `https://arxiv.org/abs/1910.02054`.

[24] Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8694–8705, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.481. URL `https://aclanthology.org/2024.naacl-long.481`.

[25] Jan Schneider, Pierre Schumacher, Simon Guist, Le Chen, Daniel Häufle, Bernhard Schölkopf, and Dieter Büchler. Identifying policy gradient subspaces, 2024. URL `https://arxiv.org/abs/2401.06604`.

[26] Namrata Vaswani, Thierry Bouwmans, Sajid Javed, and Praneeth Narayanamurthy. Robust subspace learning: Robust pca, robust subspace tracking, and robust subspace recovery. *IEEE Signal Processing Magazine*, 35(4):32–55, July 2018. ISSN 1558-0792. doi: 10.1109/msp.2018.2826566. URL `http://dx.doi.org/10.1109/MSP.2018.2826566`.

[27] Wenhan Xia, Chengwei Qin, and Elad Hazan. Chain of lora: Efficient fine-tuning of language models via residual learning, 2024. URL `https://arxiv.org/abs/2401.04151`.

[28] Can Yaras, Peng Wang, Wei Hu, Zhihui Zhu, Laura Balzano, and Qing Qu. Invariant low-dimensional subspaces in gradient descent for learning deep matrix factorizations. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023.

[29] Can Yaras, Peng Wang, Laura Balzano, and Qing Qu. Compressible dynamics in deep overparameterized low-rank learning & adaptation. *arXiv preprint arXiv:2406.04112*, 2024.

[30] Dejiao Zhang and Laura Balzano. Global convergence of a grassmannian gradient descent algorithm for subspace estimation, 2016. URL `https://arxiv.org/abs/1506.07405`.

[31] Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more, 2024. URL `https://arxiv.org/abs/2406.16793`.

[32] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection, 2024. URL `https://arxiv.org/abs/2403.03507`.

A Fine-Tuning RoBERTa-Base

To compare the computational efficiency and performance of SubTrack with GaLore, we fine-tuned RoBERTa-Base using the hyperparameters reported in Table 2, which are identical to those reported in the GaLore paper for rank-8 subspaces, with a subspace update interval of 500 iterations.

Table 2: Hyperparameters of fine-tuning RoBERTa-Base.

	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Batch Size	16	16	16	32	16	16	16	16
# Epochs	30	30	30	30	30	30	30	30
Learning Rate	1E-05	2E-05	2E-05	1E-05	1E-05	2E-05	2E-05	3E-05
SubTrack Step Size	0.001	0.01	15.0	3.0	0.001	0.001	1.0	1.0
Rank Config.				$r = 8$				
α				2				
Max Seq. Len.				512				

B Convergence of SubTrack

Convergence of SubTrack Suppose gradient has the following form (also (1)) with functions A_i, B_i, and C_i being L-continuous with constants L_A, L_B, and L_C w.r.t. weight matrix W_t; and $\|W_t\|_F \leq M$; where W_t denotes the weight matrix at step t, and M is a scalar value,

$$G = \sum_i A_i + \sum_i B_i W C_i.$$

Now, define $\widehat{B}_{i,t} = (S^l_{i,t})^\top B_i(W_t) S^l_{i,t}$ and $\widehat{C}_{i,t} = (S^r_{i,t})^\top C_i(W_t) S^r_{i,t}$, where $S^l_{i,t}$ and $S^r_{i,t}$ are the rank-r left and right projection matrices; $B_i(W_t)$ and $C_i(W_t)$ denote the dependence of B_i and C_i on the weight matrices W_t. Further letting $P_t = {S^l_t}^\top G_t S^r_t$, and $\kappa_t = \frac{1}{N}\sum_i \lambda_{min}(\widehat{B}_{i,t})\lambda_{min}(\widehat{C}_{i,t})$, where $\lambda_{min}(\cdot)$ denotes the minimum eigenvalue over each batch, and assuming that the projection matrices remain constant during the training. Then for learning-rate μ and $min(\kappa_t) > (L_A + 2L_B L_C M^2)$, the SubTrack, with $\rho_t \equiv 1$ (the element-wise regularizer of the optimizer) satisfies:

$$\|P_t\|_F \leq [1 - \mu(\kappa_{t-1} - L_A - 2L_B L_C M^2)]\|P_{t-1}\|_F.$$

That is, $P_t \to 0$ and SubTrack converges.

proof. To demonstrate that SubTrack converges to the global minimum during training, we begin by deriving the recursive form of the gradients.

Let $\otimes$ denote the Kronecker product. Then, $vec(AXB) = (B^\top \otimes A)vec(X)$.

By applying vec to the gradient form given in the theorem, we obtain:

$$g_t = vec(G_t) = vec(\sum_i A_i + \sum_i B_i W C_i) = a_t - D_t w_t \tag{10}$$

where $g_t := vec(G_t)$, $w_t := vec(W_t)$, $a_t := \frac{1}{N}\sum_i vec(A_{i,t})$, and $D_t = \frac{1}{N}\sum_i C_{i,t} \otimes B_{i,t}$.

As defined in the theorem, let $P_t = {S^l_t}^\top G_t S^r_t$. Its vectorized form can be expressed using the Kronecker product as follows:

$$\begin{aligned} p_t = vec(P_t) = vec({S^l_t}^\top G_t S^r_t) = ({S^r_t}^\top \otimes {S^l_t}^\top) vec(G_t) \\ = (S^r_t \otimes S^l_t)^\top vec(G_t) = (S^r_t \otimes S^l_t)^\top g_t \end{aligned} \tag{11}$$

Now recalling $\widehat{G}_t$ from (9), it can be written as:

$$\widehat{G}_t = S^l_t {S^l_t}^\top G_t S^r_t {S^r_t}^\top$$

Thus, its vectorized form will be:

$$\begin{aligned} vec(\widehat{G}_t) = \widehat{g}_t = vec(S_t^l {S_t^l}^\top G_t S_t^r {S_t^r}^\top) = vec(S_t^l P_t {S_t^r}^\top) \\ = (S_t^r \otimes S_t^l) vec(P_t) = (S_t^r \otimes S_t^l) p_t \end{aligned} \tag{12}$$

This is where the constant subspace assumption becomes necessary. To derive the recursive form of g_t, we assume that the projection matrices remain fixed throughout training, i.e., $S_t^r = S^r$ and $S_t^l = S^l$. Consequently, we can restate equations (11) and (12) as follows:

$$p_t = (S^r \otimes S^l)^\top g_t \tag{13}$$

$$\widehat{g}_t = (S^r \otimes S^l) p_t \tag{14}$$

Then we can write the recursive form of g_t:

$$\begin{aligned} g_t = a_t - D_t w_t = (a_t - a_{t-1}) + (D_{t-1} - D_t) w_t + a_{t-1} - D_{t-1} w_t \\ = e_t + a_{t-1} - D_{t-1}(w_{t-1} + \mu \widehat{g}_{t-1}) = e_t + g_{t-1} - \mu D_{t-1} \widehat{g}_{t-1} \end{aligned} \tag{15}$$

where $e_t := (a_t - a_{t-1}) + (D_{t-1} - D_t) w_t$. To obtain p_t from this recursive formulation, we can left-multiply by $(S^r \otimes S^l)^\top$, as shown in (14):

$$p_t = (S^r \otimes S^l)^\top e_t + (S^r \otimes S^l)^\top g_{t-1} - \mu (S^r \otimes S^l)^\top D_{t-1} \widehat{g}_{t-1} \tag{16}$$

Now, based on (13) and (14), p_t can be written as:

$$p_t = (S^r \otimes S^l)^\top e_t + p_{t-1} - \mu (S^r \otimes S^l)^\top D_{t-1} (S^r \otimes S^l) p_{t-1} \tag{17}$$

Let define:

$$\begin{aligned} \widehat{D}_t := (S^r \otimes S^l)^\top D_t (S^r \otimes S^l) = \frac{1}{N} \sum_i (S^r \otimes S^l)^\top (C_{i,t} \otimes B_{i,t})(S^r \otimes S^l) \\ = \frac{1}{N} \sum_i ({S^r}^\top C_{i,t} S^r) \otimes ({S^l}^\top B_{i,t} S^l) \end{aligned} \tag{18}$$

Then we can expand (17) and show that:

$$p_t = (I - \mu \hat{D}_{t-1}) p_{t-1} + (S^r \otimes S^l)^\top e_t \tag{19}$$

Note that S^l and S^r are orthonormal matrices. This is ensured because the subspace is initialized using the SVD of G_0, and the Grassmannian update rule provided in (7) preserves the orthonormality of the subspace matrices throughout training. Since S^l and S^r are orthonormal, we have ${S^l}^\top S^l = I$ and ${S^r}^\top S^r = I$. Consequently, we can bound the norm of the second term in (19) as follows:

$$\|(S^r \otimes S^l)^\top e_t\|_2 = \|vec({S^l}^\top E_t S^r)\|_2 = \|{S^l}^\top E_t S^r\|_F \leq \|E_t\|_F \tag{20}$$

Here E_t is the matrix form of e_t, and as declared before, $e_t := (a_t - a_{t-1}) + (D_{t-1} - D_t) w_t$, thus:

$$E_t := \frac{1}{N} \sum_i (A_{i,t} - A_{i,t-1}) + \frac{1}{N} \sum_i (B_{i,t-1} W_t C_{i,t-1} - B_{i,t} W_t C_{i,t}) \tag{21}$$

Next, we need to find an upper bound for the norm of each term in (21) to establish an upper bound for $\|E_t\|_F$. Based on the assumptions of the theorem, A_i, B_i, and C_i exhibit L-Lipschitz continuity with constants L_A, L_B, and L_C, respectively. Additionally, $\|W_t\|_F$ is bounded by a scalar M. We have:

$$\|A_t - A_{t-1}\|_F \quad \leq \quad L_A \|W_t - W_{t-1}\|_F = \mu L_A \|\tilde{G}_{t-1}\|_F \leq \mu L_A \|P_{t-1}\|_F \tag{22}$$

In the first equality, we apply (8), while the last equality holds due to (14) and the orthonormality of the projection matrices. The subsequent two inequalities can be derived similarly using these equations.

$$\begin{aligned} \|(B_t - B_{t-1}) W_t C_{t-1}\|_F \leq L_B \|W_t - W_{t-1}\|_F \|W_t\|_F \|C_{t-1}\|_F \\ = \mu L_B L_C M^2 \|P_{t-1}\|_F \end{aligned} \tag{23}$$

$$\begin{aligned}\|B_t W_t (C_{t-1} - C_t)\|_F &\le L_C \|B_t\|_F \|W_t\|_F \|W_{t-1} - W_t\|_F \\ &= \mu L_B L_C M^2 \|P_{t-1}\|_F\end{aligned} \tag{24}$$

We can now derive the bound for $\|E_t\|_F$ as follows:

$$\begin{aligned}\|E_t\|_F \le \mu L_A \|\tilde{G}_{t-1}\|_F &\le \mu L_A \|P_{t-1}\|_F + \mu L_B L_C M^2 \|P_{t-1}\|_F + \mu L_B L_C M^2 \|P_{t-1}\|_F \\ &= \mu(L_A + 2 L_B L_C M^2)\|P_{t-1}\|_F\end{aligned} \tag{25}$$

To calculate the norm bound for the first term in (19), we first need to establish the bounds for $\widehat{D}_t$. This involves estimating the minimum eigenvalue of $\widehat{D}_t$.

If we define $\gamma_{min,i,t} = \lambda_{min}(S^{l\top} B_{i,t} S^l)\lambda_{min}(S^{r\top} C_{i,t} S^r)$, then it follows that $\lambda_{min}((S^{l\top} B_{i,t} S^l) \otimes (S^{r\top} C_{i,t} S^r)) = \gamma_{min,i,t}$. Consequently, $\widehat{D}_t$ will satisfy the following inequality for every unit vector :

$$^\top \widehat{D}_t = \frac{1}{N} \sum_i^\top \left[(S^{l\top} B_{i,t} S^l) \otimes (S^{r\top} C_{i,t} S^r)\right] \ge \frac{1}{N} \sum_i \gamma_{min,i,t} \tag{26}$$

this actually provides a lower bound for eigenvalues of $\widehat{D}_t$, thus:

$$\lambda_{\max}(I - \mu \widehat{D}_{t-1}) \le 1 - \frac{\mu}{N} \sum_i \gamma_{min,i,t-1} \tag{27}$$

considering the definition of κ_t in the theorem, we can now easily show that:

$$\|P_t\|_F \le [1 - \mu(\kappa_{t-1} - L_A - 2 L_B L_C M^2)]\|P_{t-1}\|_F.$$

and completing the proof.

C Grassmann Exponential

Grassmann Exponential Let $P = UU^T \in \text{Gr}(n,p)$ be a point on the Grassmannian manifold, where $U \in \text{St}(n,p)$ is the orthonormal basis of the corresponding subspace. Consider a tangent vector $\Delta \in T_P\text{Gr}(n,p)$, and let Δ_U^{hor} denote the horizontal lift of Δ to the horizontal space at U in the Stiefel manifold $\text{St}(n,p)$. Suppose the thin SVD of Δ_U^{hor} is given by $\Delta_U^{\text{hor}} = \hat{Q}\Sigma V^T$, where $\hat{Q} \in \text{St}(n,r)$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$ contains the nonzero singular values of Δ_U^{hor} with $r = \min(p, n-p)$, and $V \in \text{St}(p,r)$. The Grassmann exponential map, representing the geodesic emanating from P in the direction Δ, is given by:

$$\text{Exp}_P^{\text{Gr}}(t\Delta) = \left[UV\cos(t\Sigma)V^T + \hat{Q}\sin(t\Sigma)V^T + UV_\perp V_\perp^T\right],$$

where $V_\perp \in \mathbb{R}^{p \times (p-r)}$ is any orthogonal complement of V.

proof. Using Grassmannina mathematics, we know that every $\Delta \in T_P Gr(n,p)$ is of the form

$$\Delta = Q \begin{pmatrix} 0 & B^T \\ B & 0 \end{pmatrix} Q^T = \left[Q \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} Q^T, P\right] \tag{28}$$

Then the lift of $\Delta \in T_P Gr(n,p)$ to $Q = (U \quad U_\perp)$ can also be calculated explicitly as follows:

$$\Delta_Q^{\text{hor}} = [\Delta, P]Q = Q \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \tag{29}$$

To resume our proof, we need to define the orthogonal group and specifying its tangent space.

Orthogonal Group The orthogonal group $O(n)$ is defined as the set of all $n \times n$ matrices Q over $\mathbb{R}$ such that $Q^T Q = QQ^T = I_n$, where Q^T is the transpose of Q and I_n is the $n \times n$ identity matrix:

$$O(n) = \{Q \in \mathbb{R}^{n \times n} \mid Q^T Q = I_n = QQ^T\}$$

Then the tangent space of the orthogonal group $O(n)$ at a point Q, denoted $T_QO(n)$, is defined as the set of matrices of the form $Q\Omega$, where $\Omega \in \mathbb{R}^{n\times n}$ is a skew-symmetric matrix, i.e., $\Omega^T = -\Omega$:

$$T_QO(n) = \{Q\Omega \mid \Omega \in \mathbb{R}^{n\times n}, \Omega^T = -\Omega\}.$$

The geodesic from $Q \in O(n)$ in direction $Q\Omega \in T_QO(n)$ is calculated via

$$\mathrm{Exp}_Q^O(tQ\Omega) = Q \exp_m(t\Omega), \tag{30}$$

If $P \in Gr(n,p)$ and $\Delta \in T_PGr(n,p)$ with $\Delta_Q^{\text{hor}} = Q\begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix}$, the geodesic in the Grassmannian is therefore

$$\mathrm{Exp}_P^{Gr}(t\Delta) = \pi^{OG}\left(Q\exp_m\left(t\begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix}\right)\right). \tag{31}$$

where π^{OG} is the projection from $O(n)$ to $Gr(n,p)$. If the thin SVD of B is given by

$$B = U_\perp^T \Delta_U^{\text{hor}} = U_\perp^T \hat{Q}\Sigma V^T$$

with $W := U_\perp^T\hat{Q} \in St(n-p,r), \Sigma \in \mathbb{R}^{r\times r}, V \in St(p,r)$. Let $W_\perp, V_\perp$ be suitable orthogonal completions. Then,

$$\exp_m\begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} V & V_\perp & 0 & 0 \\ 0 & 0 & W & W_\perp \end{pmatrix}\begin{pmatrix} \cos(\Sigma) & 0 & -\sin(\Sigma) & 0 \\ 0 & I_{p-r} & 0 & 0 \\ \sin(\Sigma) & 0 & \cos(\Sigma) & 0 \\ 0 & 0 & 0 & I_{n-p-r} \end{pmatrix}\begin{pmatrix} V^T & 0 \\ V_\perp^T & 0 \\ 0 & W^T \\ 0 & W_\perp^T \end{pmatrix},$$

which leads to the desired result when inserted into (31). For more mathematical details, you can refer to Edelman et al. [8], Bendokat et al. [3], or other useful resources on Grassmann geometry.