NEURAL INFORMATION PROCESSING SYSTEMS

Enhancing Fine-Tuning Efficiency of LLMs Through Gradient Subspace Tracking

Sahar Rajabi, Sirisha Rambhatla

Management Science and Engineering, University of Waterloo, Waterloo, Ontario, Canada

Critical ML

Overview of SubTrack

SubTrack enhances the computational efficiency of training and fine-tuning LLMs while minimizing memory requirements. Rather than targeting weight matrices, SubTrack focuses on optimizing gradient matrices and reducing the memory footprint of optimizer states. The approach begins by projecting gradients into a core subspace, effectively compressing the optimizer's states. This is followed by leveraging the Grassmannian manifold's geometry to dynamically track the subspace using rank-r computations, ensuring minimal computational overhead.

1. Motivation

- LLMs demonstrate state-of-the-art performance across various tasks and are increasingly popular.
- Training LLMs is resource-intensive, limiting accessibility and increasing the carbon footprint.
- Optimizer states account for a large share of memory; reducing them can significantly cut resource usage.
- Existing approaches either partially optimize the network or introduce high computational overhead.
- Can we achieve memory reduction with full-parameter tuning while minimizing runtime impact?

3. SubTrack: Tracking the Gradient Subspace

Preliminaries

- The **Grassmannian manifold** $Gr(n,p)$ parametrizes the set of all p-dimensional subspaces of $\mathbb{R}^n$ [3,4].
- **Geodesics** are the shortest path between two points on a manifold.
- Gradients with the following form will lead to a low-rank gradient during optimization with high probability [1]. This fact has been considered in GaLore [1] as well.

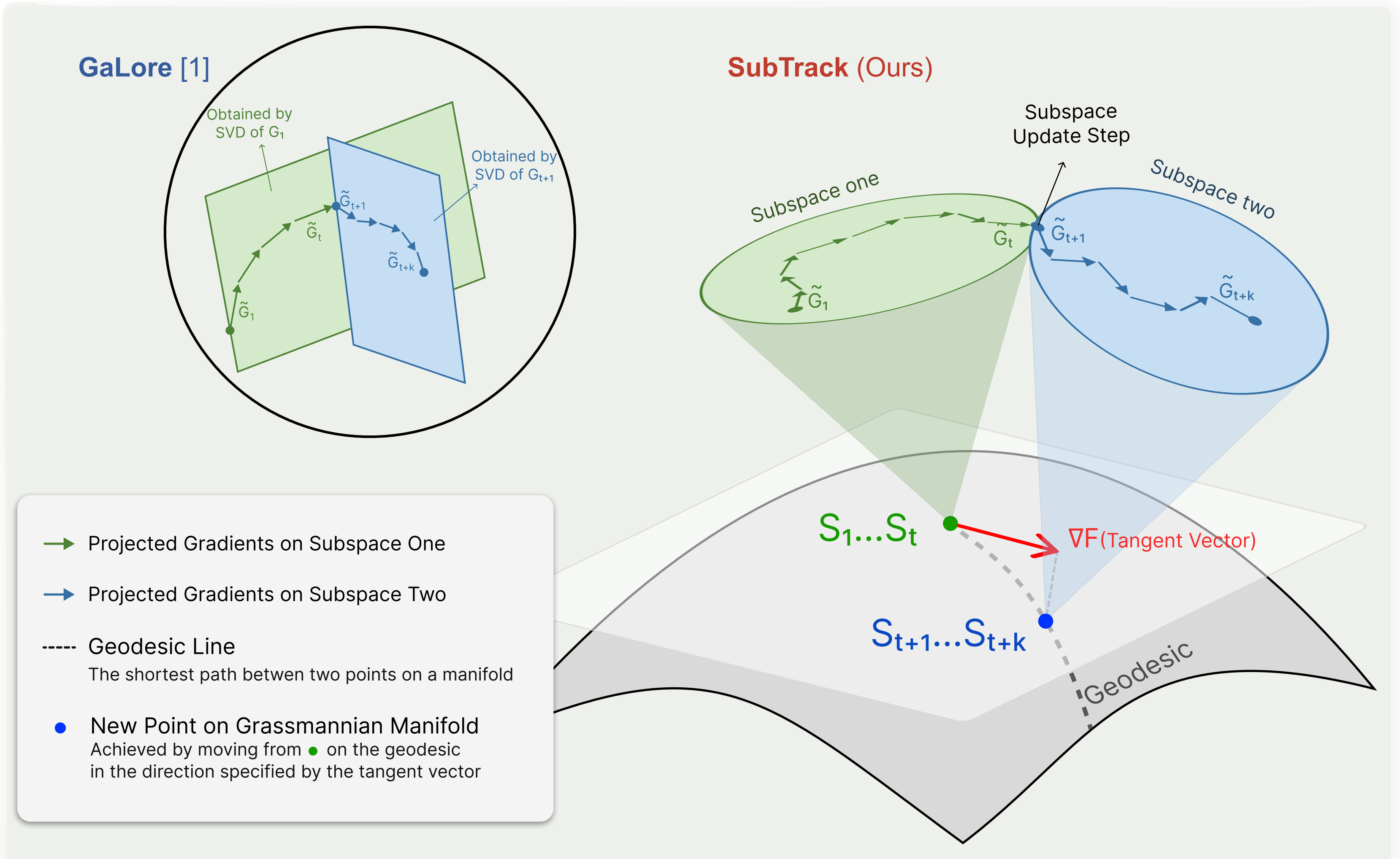
$$G = \sum_i A_i + \sum_i B_i W C_i \qquad (1)$$

(G: $m \times n$ gradient matrix | W: $m \times n$ weight matrix | A_i: constant $m \times n$ matrix)

(B_i: PSD $m \times m$ matrix | C_i: PSD $n \times n$ matrix | i: index in the batch)

The presented form is applicable to all linear layers like query, key, and value of attention modules, or MLP layers in the network. It also applies to a more general non-linear networks known as reversible networks [1].

How does SubTrack differ from GaLore?


GaLore [1]
Obtained by SVD of G₁
Obtained by SVD of Gₜ₊₁
SubTrack (Ours)
Subspace Update Step
Subspace one
Subspace two
Projected Gradients on Subspace One
Projected Gradients on Subspace Two
Geodesic Line
The shortest path betwen two points on a manifold
New Point on Grassmannian Manifold
Achieved by moving from ● on the geodesic in the direction specified by the tangent vector
S₁...Sₜ
∇F(Tangent Vector)
Sₜ₊₁...Sₜ₊ₖ
Geodesic

Visualization of SubTrack: Between subspace update steps, gradients are projected onto a fixed subspace. The tangent vector ∇F is computed via the derivative of a loss function that measures the subspace estimation error between updates. The subspace is then updated by moving along the corresponding geodesic, determined by ∇F, on the Grassmannian manifold to minimize the measured error.

Visualization of GaLore: Between subspace update steps, gradients are projected onto a fixed subspace. The subspace is then updated by performing SVD on the latest observed gradient matrix.

Methodology

1. Subspace initialization via SVD; our first and last heavy computation!

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

(G_0: $m \times n$ gradient matrix at step 0 | U, S, and V: SVD components of G_0 | r: subspace rank)

- Left projection if $m \le n$; right projection otherwise, to further reduce memory!
- Assume $m \le n$ w.l.o.g. in the rest of presentation.

2. Accumulated gradient can be used to reduce noise-effect (further experiments shows that SubTrack achieves on par result using only G_t that leads to save more memory).

$$G_{acc} = \frac{1}{T_n - T_{n-1}} \sum_{t=T_{n-1}}^{T_n} G_t \qquad (3)$$

(T_n and T_{n-1}: indices of two consecutive subspace update steps)

3. Adjust subspace estimation in subspace update steps [3, 4].

3.1. Following cost function is used to minimize the subspace (S_t) estimation error.

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

(A: solution to the least squares problem | S_t: represents current subspace)

3.2. Computing residual as $R = G_{acc} - S_t A$

3.3. Computing tangent vector ∇F on Grassmannian manifold:

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

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

3.4. Compute rank-1 estimation of ∇F using its SVD, denoted by $\hat{U}_F \hat{\Sigma}_F \hat{V}_F^\top$

3.5. Update the underlying subspace by moving along the Grassmannian manifold [3, 4].

$$S_{t+1}(\eta) = \begin{pmatrix} S_t \hat{V}_F & \hat{U}_F \end{pmatrix} \begin{pmatrix} \cos \hat{\Sigma}_F \eta \\ \sin \hat{\Sigma}_F \eta \end{pmatrix} \hat{V}_F^\top + S_t \left(I - \hat{V}_F \hat{V}_F^\top\right) \qquad (7)$$

This update step preserves the orthonormality of S_{t+1}, ensuring it remains on the Grassmannian manifold.

Acknowledgment

NSERC CRSNG

Sirisha Rambhatla would like to acknowledge support of the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, RGPIN-2022-03512.

2. Prior Works

Memory Efficiency

- LoRA [5] reduces weight space by decomposing weight matrices into trainable low-rank matrices.
- BAdam [6] and BlockLLM [7] cut optimizer states by iteratively and partially optimizing the network.
- GaLore [1] enables full-parameter training by projecting gradients into a low-dimensional subspace, updating with periodic SVD. *Issues:* SVD is computationally expensive and noise-sensitive.
- Prior research like [8, 9] showed that gradients evolve in a low-dimensional subspaces in gradient descent; an observation which is employed by many research like ours and GaLore [1].

Subspace Tracking

- Subspace tracking is common in signal processing, with methods like Iterative SVD or recursive least square.
- GROUSE [2] uses Grassmannian manifolds for incremental subspace updates with partially observed data.

4. Advantages of SubTrack

- **Full-parameter optimization** while reducing memory consumption
- **Significant reduction in wall-time** compared to similar methods like GaLore
- Achieves **better or comparable performance** compared to GaLore
- Allowing **higher subspace updates frequencies** without drastic wall-time increase
- **Reducing noise effect** by leveraging the subspace estimation error as a signal to adjust the subspace

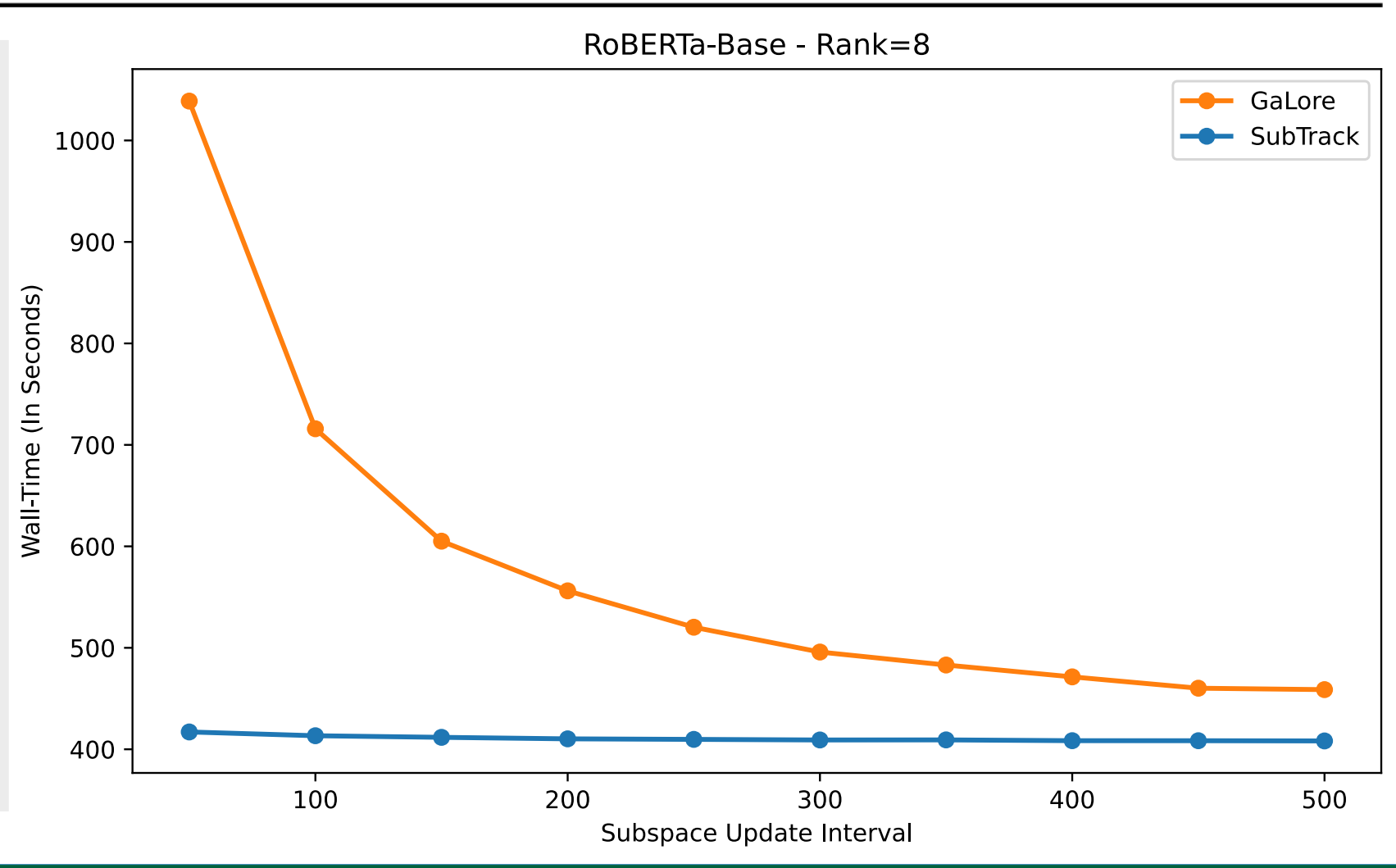
5. Experiments and Results

- Performance and wall-time on fine-tuning RoBERTa-Base on GLUE tasks with subspace of rank 4 and 8.
- Performance is reported after fine-tuning for 30 epochs.
- Wall-times are measured after 2500 iterations to include exactly 5 subspace updates steps.

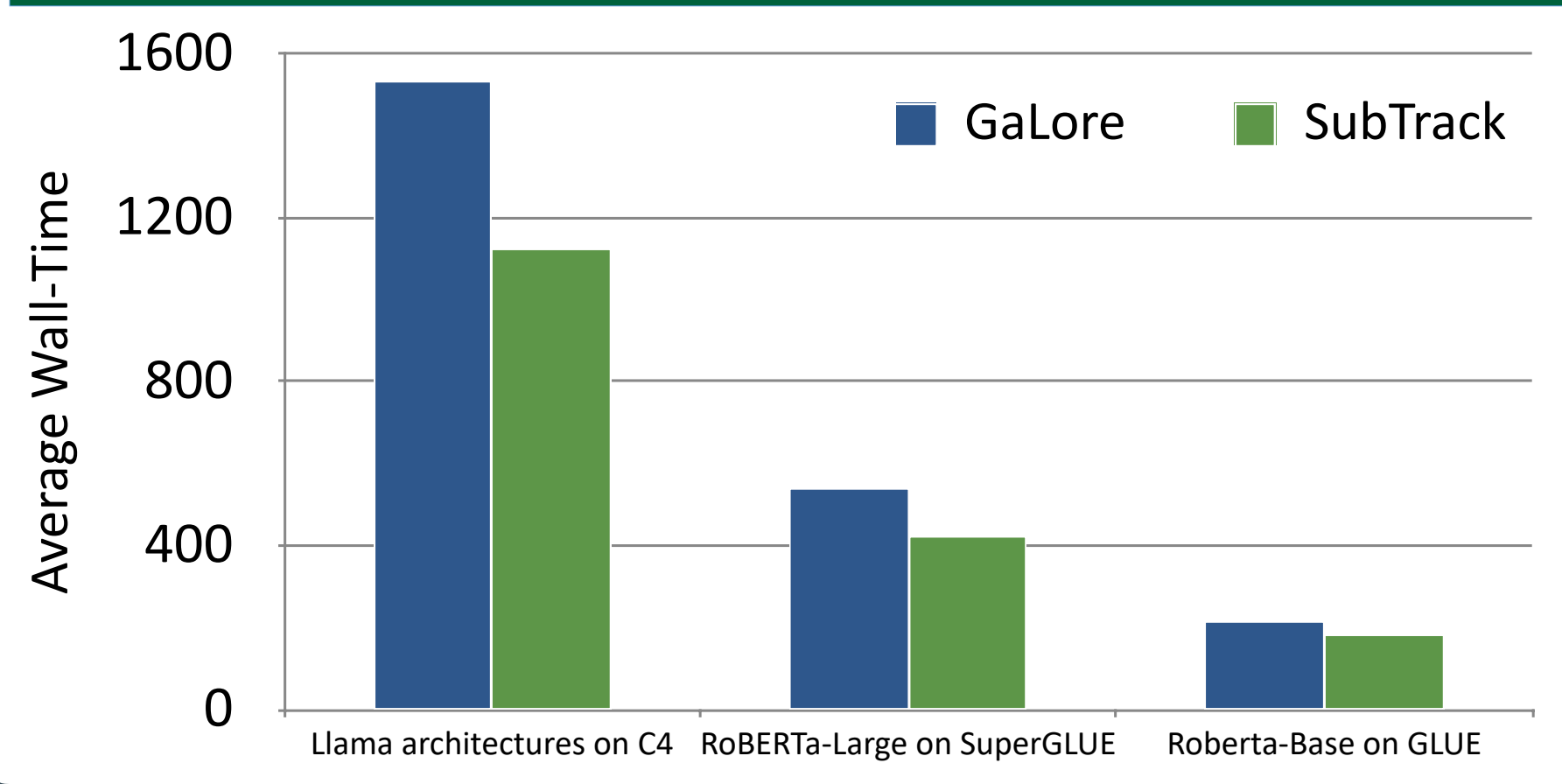
		COLA	STS-B	MRPC	RTE	SST-2	MNLI	QNLI	QQP	Avg
GaLore Rank = 4	Perf.	60.34	90.58	92.58	76.53	94.27	87.12	92.20	87.86	85.18
	Time	195.7	195.1	200.2	325.8	185.7	206.8	216.5	190.4	214.5
SubTrack (Ours) Rank = 4	Perf.	61.32	90.64	92.66	77.98	94.15	86.85	91.85	87.50	85.37
	Time	155.7	158.0	172.7	305.7	147.5	175.0	192.2	155.8	182.8
Reduction in Wall-Time		**20.44%**	**19.02%**	**13.74%**	**6.17%**	**20.57%**	**15.38%**	**11.22%**	**18.17%**	**15.59%**
GaLore Rank = 8	Perf.	58.54	90.61	91.30	74.37	94.50	87.34	92.71	87.99	84.67
	Time	188.0	195.6	196.5	328.3	187.1	208.0	217.1	189.4	213.7
SubTrack (Ours) Rank = 8	Perf.	58.54	90.87	91.43	76.53	94.27	87.09	92.49	87.57	84.85
	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%**

Run-Time Consistency

- RoBERTa-Base is fine-tuned for 10 epochs on COLA.
- Subspace update intervals range from 50 to 500.
- By increasing subspace update interval, the update frequency actually decreases
- SubTrack demonstrates superior run-time consistency compare to GaLore, when increasing subspace update frequency.



Preview of Some New Results



Wall-Time Reduction

- Fine-Tuning RoBERTa-Base on GLUE tasks
  - **15%** on average (up to **20.5%**)
- Fine-Tuning RoBERTa-Large on SuperGLUE tasks
  - **22%** on average (up to **65%**)
- Pre-Training Llama-based architectures on C4
  - **27%** on average (up to **49%** on 3B model size)
  - Size ranges from 60M to 3B

Algorithm

- The following shows the subspace update steps algorithm. Projected gradient is passed to the optimizer, and the optimizer's output will be projected back to high-dimensional space for a completely usual back-propagation, which completely **preserves training dynamics**.

Algorithm 1 SubTrack

```
Require: Sequence of $m \times n$ gradients $G_t$ with $m \le n$ (w.l.o.g.), step-size $\eta$, rank $r$, subspace update steps k
  Initialize Subspace via SVD Decomposition:
  $P_0 \leftarrow U[:, :r]$ , where $U, S, V \leftarrow \mathrm{SVD}(G_0)$
  $S_0 \leftarrow P_0$                                   {The initial subspace}
  $G_{acc} = 0_{m \times n}$                             {To keep the accumulated gradient}
  for $t = 1, \dots, T$ 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: $S_t^\top G_t$
```

References

[1] Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., & Tian, Y. (2024). GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. Forty-First International Conference on Machine Learning (ICML).

[2] Balzano, L., Nowak, R., & Recht, B. (2010). Online identification and tracking of subspaces from highly incomplete information. 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 704–711.

[3] Bendokat, T., Zimmermann, R., & Absil, P.-A. (2024). A grassmann manifold handbook: Basic geometry and computational aspects. *Advances in Computational Mathematics*, *50*(1).

[4] Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, *20*(2).

[5] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., … Chen, W. (2022). LoRA: Low-Rank Adaptation of Large Language Models. International Conference on Learning Representations (ICLR).

[6] Luo, Q., Yu, H., & Li, X. (2024). BAdam: A Memory Efficient Full Parameter Optimization Method for Large Language Models. The Thirty-Eighth Annual Conference on Neural Information Processing Systems (NeurIPS).

[7] Ramesh, A. V., Ganapathiraman, V., Laradji, I. H., & Schmidt, M. (2024). BlockLLM: Memory-Efficient Adaptation of LLMs by Selecting and Optimizing the Right Coordinate Blocks.

[8] Yaras, C., Wang, P., Hu, W., Zhu, Z., Balzano, L., & Qu, Q. (2023). Invariant Low-Dimensional Subspaces in Gradient Descent for Learning Deep Matrix Factorizations.

[9] Gur-Ari, G., Roberts, D. A., & Dyer, E. (2018). Gradient descent happens in a tiny subspace. arXiv Preprint arXiv:1812. 04754.