SubTrack++ : Gradient Subspace Tracking for Scalable LLM Training

Sahar Rajabi, Nayeema Nonta, Sirisha Rambhatla

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

Overview of SubTrack++

We propose **SubTrack++**, a method that achieves state-of-the-art convergence by exploiting Grassmannian geometry for efficient training. SubTrack++ performs **Gradient Subspace Tracking** combined with **projection-aware optimizers** to ensure Adam's internal statistics adapt to subspace changes. It further incorporates a **recovery scaling** technique to restore information lost during low-rank projections. Our approach significantly enhances efficiency, **reducing pre-training wall-time by up to 65%** and **fine-tuning time by 36%** compared to existing SOTA methods, all while maintaining the same memory footprint.

1. Motivation

- LLM training is bottlenecked by the massive parameter count and significant optimizer state memory overhead.
- Current efficiency methods force compromises between memory, training time, and final performance; our goal is simultaneous advancement across all three dimensions.
- The core motivation stems from the observation that LLM gradients reside in a persistently low-rank subspace [8].
- Standard optimizers like Adam lose their ability to track the necessary internal statistics for effective adaptation when the subspace changes. SubTrack++ overcomes this by using projection-aware optimizers [4].
- Low-rank projections cause information lost; we counteract this with a recovery scaling technique [2, 3].
- SubTrack++ utilizes Grassmannian subspace tracking [6, 7] to robustly and efficiently manage the evolution of the low-dimensional subspace, providing a unified, principled foundation for gradient projection in LLMs.

2. Prior Works

- The recent low-rank method have reduced the memory footprint of training and fine-tuning LLMs, by exploiting the low-dimensional nature of weights and gradients and projecting them into low-rank subspaces.
- GaLore [1] projects gradients into a low-rank subspace and adjusting the subspace using periodic SVD.
- Fira [2] enhances GaLore [1] via norm-based scaling to reintegrate information lost during low-rank projection.
- APOLLO [3] uses channel-wise learning rate via random low-rank projections, and recovering the lost information.
- LDAdam [4] performs adaptive optimization in low-dimensional subspaces using PowerSGD-based iterative updates, incorporating a projection-aware optimizer and an error-feedback mechanism.
- Online Subspace Descent [5] reduces complexity by employing Online PCA for dynamically tracking and updating the projection matrix, avoiding expensive SVD.

3. SubTrack++: Gradient Subspace Tracking for Scalable LLM Training

- The core of low-rank gradient methods is the projection of the full gradient matrix onto a low-rank subspace to reduce the optimizer's memory footprint, followed by projecting the processed gradient back for the weight update.
- Let $G_t \in \mathbb{R}^{m\times n}$ be the full gradient matrix at iteration t, and r be the desired low-rank dimension ($r \ll min(m,n)$); The gradient is then projected into a low-rank subspace, spanned by an orthonormal basis matrix $S_t \in \mathbb{R}^{m\times r}$. Assuming $m \le n$ (without loss of generality, to minimize memory, methods choose the left or right singular vectors depending on dimensions [1]).
- The projected gradient $\widetilde{G}_t$ is computed by projecting G_t onto the current subspace S_t: $\widetilde{G}_t = S_t^\top G_t \in \mathbb{R}^{r\times n}$. The low-rank matrix $\widetilde{G}_t$ is what the optimizer operates on, drastically reducing the optimizer's state memory footprint.
- The resulting low-rank output of the optimizer, $\widehat{G}_t^O$, is then projected back to the full parameter space for the final weight update.

Algorithm 1 SubTrack++
(Subspace Tracking , Projection-Aware Optimizer , Recovery Scaling , Regular Adam)

Require: $W_t, G_t \in \mathbb{R}^{m\times n}$ with $m \le n$ (w.l.o.g.), learning rate α, decay rates β_1 and β_2, SubTrack++ step-size η, rank r, subspace update interval k, recovery scaling limiter factor ζ. We use $\oslash$ to denote Hadamard division.

$S_0 \leftarrow U[:,:r]$, where $U,S,V \leftarrow \text{SVD}(G_0)$ {Initializing First Subspace}
for $t = 0,\dots,T$ **do**
  if $t \bmod k == 0$ **then**
    $G_{lr} = \arg\min_A \|(S_{t-1}A - G_t)\|^2$, and $R = G_t - S_{t-1}G_{lr}$ **(I)**
    $\nabla F = -2RG_{lr}^\top \approx \widehat{U}_F\widehat{\Sigma}_F\widehat{V}_F^\top$ **(II)**
    $S_t = (S_{t-1}\widehat{V}_F \quad \widehat{U}_F)\begin{pmatrix}\cos\widehat{\Sigma}_F\eta \\ -\sin\widehat{\Sigma}_F\eta\end{pmatrix}\widehat{V}_F^\top + S_{t-1}(I - \widehat{V}_F\widehat{V}_F^\top)$ **(III)**
    $M_t \leftarrow \beta_1\cdot(S_t^\top S_{t-1}M_{t-1}) + (1-\beta_1)\cdot\widetilde{G}_t$ **(IV)** {$\widetilde{G}_t = S_t^\top G_t$: low-rank projection of G_t}
    $\mathcal{V}_t \leftarrow \beta_2\cdot[(1-\beta_2^{t-1})|(S_t^\top S_{t-1})^2\cdot(\mathcal{V}_{t-1} - M_{t-1}^2) + (S_t^\top S_{t-1}\cdot M_{t-1})^2|] + (1-\beta_2)\cdot\widetilde{G}_t^2$ **(V)**
  else
    $S_t = S_{t-1}$
    $M_t \leftarrow \beta_1\cdot M_{t-1} + (1-\beta_1)\cdot\widetilde{G}_t$
    $\mathcal{V}_t \leftarrow \beta_2\cdot\mathcal{V}_{t-1} + (1-\beta_2)\cdot\widetilde{G}_t^2$
  end if
  $\widetilde{G}_t^O = M_t \oslash \sqrt{\mathcal{V}_t + \epsilon}$, $\widehat{G}_t = S_t\widetilde{G}_t^O$ {$\widetilde{G}_t^O$: optimizer's output, $\widehat{G}_t$: projected-back gradients }
  $\phi_t(G_t)_i = \frac{\|\widetilde{G}^O_{t,:,i}\|}{\|\widetilde{G}_{t,:,i}\|}$, $\Lambda_t = \phi_t(G_t)(G_t - S_t\widetilde{G}_t)$ **(VI)** {We use $\oslash$ to denote Hadamard division.}
  if $\frac{\Lambda_t}{\Lambda_{t-1}} > \zeta$ then $\Lambda_t \leftarrow \frac{\Lambda_t}{\|\Lambda_t\|}\cdot\zeta\|\Lambda_{t-1}\|$ **(VII)**
  $W_t \leftarrow W_{t-1} - \alpha\cdot\widehat{G}_t - \alpha\cdot\Lambda_t$ **(VIII)**
end for

Grassmannian Subspace Tracking

- The gradient subspace is not static. Accurately tracking its shifts is crucial for performance [1-5].
- We frame subspace estimation as finding a point on Grassmannian, the space of all r-dimensional subspaces in an n-dimensional space [7].
- Instead of recomputing the subspace, it uses the subspace estimation error **(I)** to compute a tangent vector ∇F **(II)** on the manifold. The subspace is then moved along its geodesic to minimize the subspace estimation error **(III)** [6, 7].
- To ensure stability, SubTrack++ uses a rank-1 approximation of ∇F **(II)**, determined by its largest singular value, to guide the update.

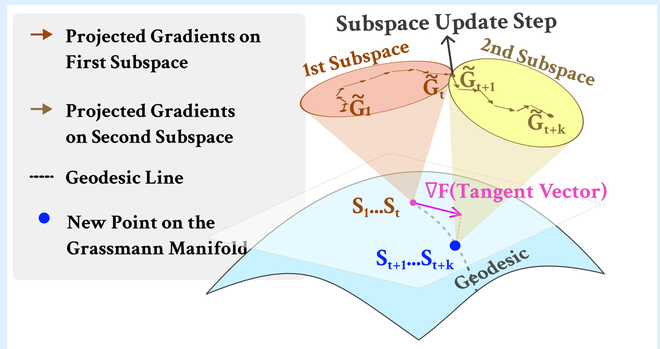


Figure 1: Visualization of Grassmannian subspace tracking: Between subspace updates, gradients are projected onto a fixed subspace. The tangent vector ∇F is computed via the derivative of a loss function, measuring the subspace estimation error. The subspace is then updated by moving along the corresponding geodesic, determined by ∇F to minimize estimation error.

Projection-Aware Optimizer

- Adam's standard update rules assume a fixed coordinate system.
- When the orthonormal basis is updated from S_{t-1} to S_t, the optimizer's old states are stored in the coordinate system defined by S_{t-1}. Applying a projected gradient defined by S_t to the old states causes a misalignment.
- This is solved by explicitly transforming the previous momentum estimates, as shown in equations **(IV)** and **(V)**, to align with the new subspace before computing the current update [4].

Recovery Scaling

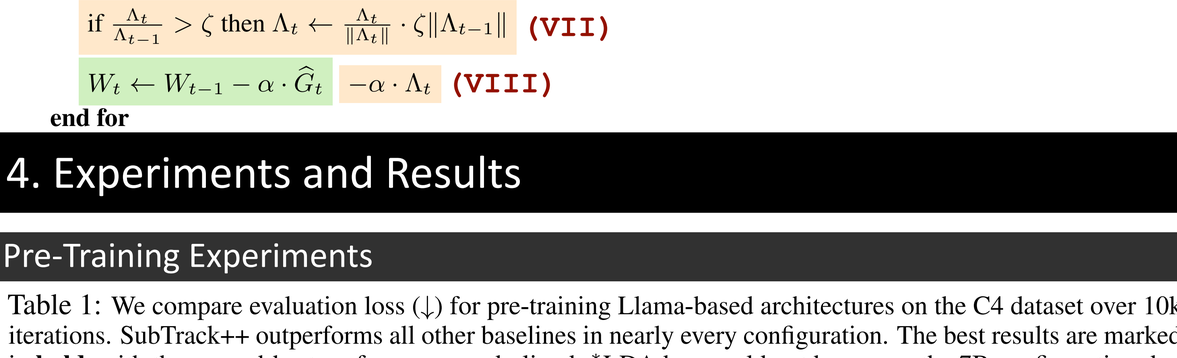
- During low-rank projection, the discarded component of the gradient can be used to boost performance [2, 3].
- Optimizers like Adam exhibit a consistent scaling behavior between low-rank and full-rank gradient regimes [2].
- A column-wise factor is computed via the norms of the optimizer's output and the input low-rank gradient **(VI)**.
- For robustness and stability, a clipping mechanism is used to limit the growth of the recovered term **(VII)**.
- The scaled term is added to the standard weight update rule **(VIII)**, ensuring the final weight update accounts for both the optimized low-rank component and the recovered information from the orthogonal complement [2].

4. Experiments and Results

Pre-Training Experiments

Table 1: We compare evaluation loss (↓) for pre-training Llama-based architectures on the C4 dataset over 10k iterations. SubTrack++ outperforms all other baselines in nearly every configuration. The best results are marked in **bold**, with the second-best performance underlined. *LDAdam could not be run on the 7B configuration due to an out-of-memory error with our available resources.

	60M r=128	**130M** r=256	**350M** r=256	**1B** r=512	**3B** r=512	**7B** r=1024
Full-Rank	3.41	3.25	3.40	4.61	4.52	4.30
GaLore [Zhao et al., 2024]	4.02	3.61	3.62	6.53	6.57	5.55
BAdam [Luo et al., 2024]	7.86	7.08	7.62	7.28	7.12	6.76
Online Subspace Descent [Liang et al., 2024]	4.18	3.88	4.09	6.79	6.85	5.69
LDAdam [Robert et al., 2025]	3.52	3.44	3.67	4.70	**4.39**	OOM*
Fira [Chen et al., 2025]	3.80	3.55	3.56	6.31	6.50	6.83
SubTrack++ (Ours)	**3.43**	**3.24**	**3.29**	**4.52**	4.50	**4.63**



(a) Loss vs. Training Steps. (b) Loss vs. Relative Wall-Time.

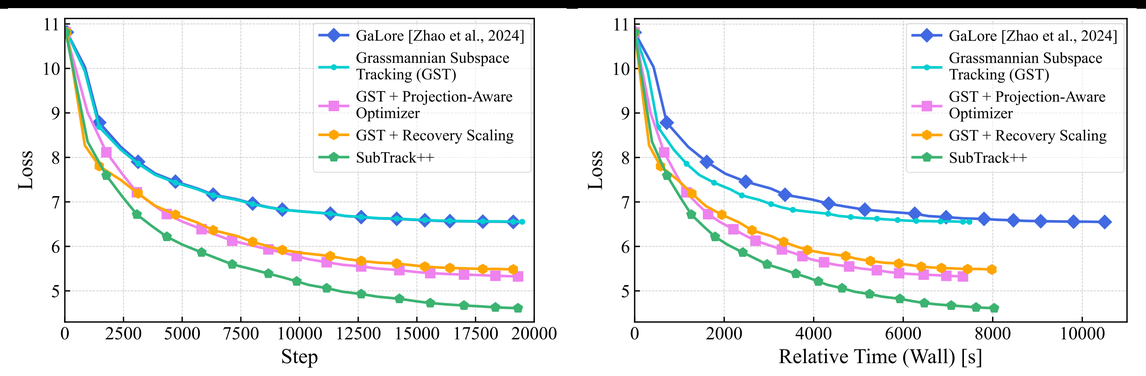
Figure 3: Comparison of baselines in pre-training Llama-1B architecture. (a) shows training loss (↓) versus training steps. (b) shows the same runs against wall-time. SubTrack++ outperforms all baselines; substantially reducing wall-time, especially compared to LDAdam, the top-performing baseline.

Fine-Tuning Experiments

Table 7: Comparing final evaluation loss (↓), wall-time, and peak memory consumption of fine-tuning Llama-2-7B-chat-hf on Alpaca dataset.

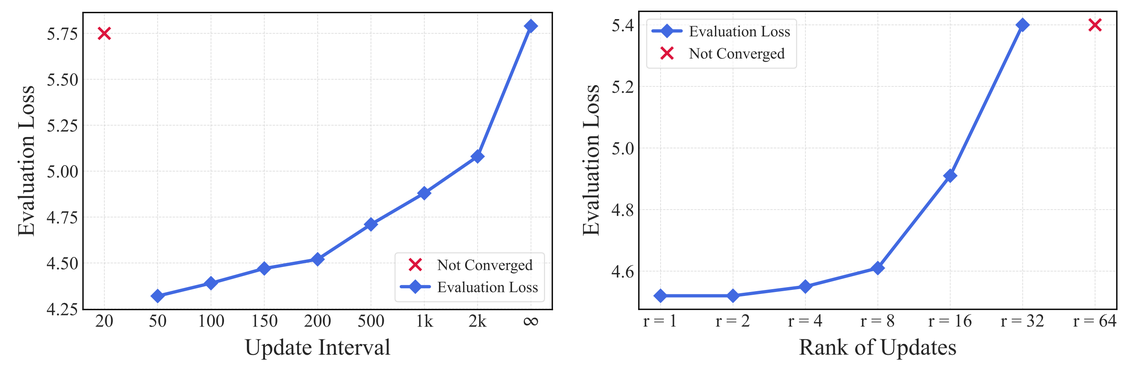
Method	**Evaluation Loss**	**Wall-Time** (mins)	**Memory** (GB)
GaLore	0.88	178	59.4
LDAdam	0.85	342	66.1
SubTrack++	0.88	117	62.5

5. Ablations



(a) Loss vs. Training Steps. (b) Loss vs. Relative Wall-Time.

Figure 4: Ablation study comparing pure Grassmannian subspace tracking with incremental additions of the projection-aware optimizer and recovery scaling, leading to SubTrack++ . While Grassmannian tracking alone almost matches GaLore's step-wise convergence (a), it significantly reduces wall-time (b).



(a) Loss vs. Subspace Update Frequency. (b) Loss vs. Subspace Update Rank.

Figure 5: Ablation results on (a) update frequency: decreasing the update interval (i.e., increasing the frequency) improves evaluation performance up to a point, but overly frequent updates hinder training convergence. (b) update rank: increasing the rank of updates degrades model performance, and beyond a certain threshold, can prevent convergence. These results emphasize the importance of controlled subspace adjustments.

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 2024).

[2] Chen, X., Fent, K., Li, C., Lai, X., Yue, X., Yuan, Y., & Wang, G. (2025). Fira: Can We Achieve Full-rank Training of LLMs under Low-rank Constraint? Thirty-Ninth Conference on Neural Information Processing Systems (NeurIPS 2025).

[3] Zhu, H., Zhang, Z., Cong, W., Liu, X., Park, S., Chandra, V., Long, B., Pan, D., Wang, Z., & Lee, J. (2025). APOLLO: SGD-Like Memory, AdamW-Level Performance. Eighth MLSys Conference.

[4] Robert, T., Safaryan, M., Modoranu, & I., Alistarh, D. (2025). LDAdam: Adaptive Optimization from Low- Dimensional Gradient Statistics. Thirty-Ninth International Conference on Learning Representations (ICLR 2025).

[5] Liang, K., Liu, B., Chen, L., & Liu, Q. (2024). Memory-Efficient LLM Training with Online Subspace Descent. Thirty-Eighth Conference on Neural Information Processing Systems (NeurIPS 2024).

[6] 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.

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

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

Acknowledgement

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