

A Convergence of SubTrack++

Theorem 3.2 (Convergence of Grassmannian Subspace Tracking). Suppose gradient has the following form with functions A_i , B_i , and C_i being L -continuous as per [Def. 3.1](#) with constants L_A , L_B , and L_C w.r.t. weight matrix W_i ; 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 $\hat{B}_{i,t} = (S_{i,t}^l)^\top B_i(W_t) S_{i,t}^l$ and $\hat{C}_{i,t} = (S_{i,t}^r)^\top C_i(W_t) S_{i,t}^r$, where $S_{i,t}^l$ and $S_{i,t}^r$ 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 $P_t = S_t^{l\top} G_t S_t^r$, and $\kappa_t = \frac{1}{N} \sum_i \lambda_{\min}(\hat{B}_{i,t}) \lambda_{\min}(\hat{C}_{i,t})$, where $\lambda_{\min}(\cdot)$ denotes the minimum eigenvalue over each batch, and N representing the number of samples in a batch. Assuming that the projection matrices remain constant during the training. Then for learning-rate μ and $\min(\kappa_t) > (L_A + 2L_B L_C M^2)$, subspace tracking, 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, $P_t \rightarrow 0$ and it 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, $\text{vec}(AXB) = (B^\top \otimes A) \text{vec}(X)$.

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

$$g_t = \text{vec}(G_t) = \text{vec}\left(\sum_i A_i + \sum_i B_i W C_i\right) = a_t - D_t w_t \quad (16)$$

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

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

$$\begin{aligned} p_t = \text{vec}(P_t) &= \text{vec}(S_t^{l\top} G_t S_t^r) = (S_t^{r\top} \otimes S_t^{l\top}) \text{vec}(G_t) \\ &= (S_t^r \otimes S_t^l)^\top \text{vec}(G_t) = (S_t^r \otimes S_t^l)^\top g_t \end{aligned} \quad (17)$$

Now recalling \hat{G}_t from [\(15\)](#), it can be written as:

$$\hat{G}_t = S_t^l S_t^{l\top} G_t S_t^r S_t^{r\top}$$

Thus, its vectorized form will be:

$$\begin{aligned} \text{vec}(\hat{G}_t) &= \hat{g}_t = \text{vec}(S_t^l S_t^{l\top} G_t S_t^r S_t^{r\top}) = \text{vec}(S_t^l P_t S_t^{r\top}) \\ &= (S_t^r \otimes S_t^l) \text{vec}(P_t) = (S_t^r \otimes S_t^l) p_t \end{aligned} \quad (18)$$

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 [\(17\)](#) and [\(18\)](#) as follows:

$$p_t = (S^r \otimes S^l)^\top g_t \quad (19)$$

$$\hat{g}_t = (S^r \otimes S^l) p_t \quad (20)$$

Then we can write the recursive form of g_t :

$$\begin{aligned} g_t &= a_t - D_t w_t = (a_t - a_{t-1}) + (D_{t-1} - D_t) w_t + a_{t-1} - D_{t-1} w_t \\ &= e_t + a_{t-1} - D_{t-1} (w_{t-1} + \mu \hat{g}_{t-1}) = e_t + g_{t-1} - \mu D_{t-1} \hat{g}_{t-1} \end{aligned} \quad (21)$$

where $e_t := (a_t - a_{t-1}) + (D_{t-1} - D_t) w_t$.

Note that in deriving (21), we utilized the general form of the weight update rule, $w_{t+1} = w_t - \mu g_t$, which can be rewritten as $w_t = w_{t+1} + \mu g_t$. By applying this rule along with (16), we arrive at the second equality in (21) as follows:

$$\begin{aligned}
g_t &= a_t - D_t w_t = a_t - D_t w_t - g_{t-1} + g_{t-1} \\
&= a_t - D_t w_t - a_{t-1} + D_{t-1} w_{t-1} + a_{t-1} - D_{t-1} w_{t-1} \\
&= a_t - D_t w_t - a_{t-1} + D_{t-1} (w_t + \mu g_{t-1}) + a_{t-1} - D_{t-1} (w_t + \mu g_{t-1}) \\
&= a_t - D_t w_t - a_{t-1} + D_{t-1} w_t + \mu D_{t-1} g_{t-1} + a_{t-1} - D_{t-1} w_t - \mu D_{t-1} g_{t-1} \\
&= a_t - a_{t-1} + (D_{t-1} - D_t) w_t + a_{t-1} - D_{t-1}
\end{aligned}$$

To obtain p_t from this recursive formulation, we can left-multiply by $(S^r \otimes S^l)^\top$, as shown in (20):

$$p_t = (S^r \otimes S^l)^\top e_t + (S^r \otimes S^l)^\top g_{t-1} - \mu (S^r \otimes S^l)^\top D_{t-1} \hat{g}_{t-1} \quad (22)$$

Now, based on (19) and (20), p_t can be written as:

$$p_t = (S^r \otimes S^l)^\top e_t + p_{t-1} - \mu (S^r \otimes S^l)^\top D_{t-1} (S^r \otimes S^l) p_{t-1} \quad (23)$$

Let define:

$$\begin{aligned}
\hat{D}_t &:= (S^r \otimes S^l)^\top D_t (S^r \otimes S^l) = \frac{1}{N} \sum_i (S^r \otimes S^l)^\top (C_{i,t} \otimes B_{i,t}) (S^r \otimes S^l) \\
&= \frac{1}{N} \sum_i (S^{r\top} C_{i,t} S^r) \otimes (S^{l\top} B_{i,t} S^l)
\end{aligned} \quad (24)$$

Then we can expand (23) and show that:

$$p_t = (I - \mu \hat{D}_{t-1}) p_{t-1} + (S^r \otimes S^l)^\top e_t \quad (25)$$

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 (5) 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 (25) as follows:

$$\|(S^r \otimes S^l)^\top e_t\|_2 = \|\text{vec}(S^{l\top} E_t S^r)\|_2 = \|S^{l\top} E_t S^r\|_F \leq \|E_t\|_F \quad (26)$$

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:

$$E_t := \frac{1}{N} \sum_i (A_{i,t} - A_{i,t-1}) + \frac{1}{N} \sum_i (B_{i,t-1} W_t C_{i,t-1} - B_{i,t} W_t C_{i,t}) \quad (27)$$

Next, we need to find an upper bound for the norm of each term in (27) 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:

$$\|A_t - A_{t-1}\|_F \leq L_A \|W_t - W_{t-1}\|_F = \mu L_A \|\tilde{G}_{t-1}\|_F \leq \mu L_A \|P_{t-1}\|_F \quad (28)$$

In the first equality, we apply (13), while the last equality holds due to (20) and the orthonormality of the projection matrices. The subsequent two inequalities can be derived similarly using these equations.

$$\begin{aligned}
\|(B_t - B_{t-1}) W_t C_{t-1}\|_F &\leq L_B \|W_t - W_{t-1}\|_F \|W_t\|_F \|C_{t-1}\|_F \\
&= \mu L_B L_C M^2 \|P_{t-1}\|_F
\end{aligned} \quad (29)$$

$$\begin{aligned}
\|B_t W_t (C_{t-1} - C_t)\|_F &\leq L_C \|B_t\|_F \|W_t\|_F \|W_{t-1} - W_t\|_F \\
&= \mu L_B L_C M^2 \|P_{t-1}\|_F
\end{aligned} \quad (30)$$

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

$$\begin{aligned}
\|E_t\|_F &\leq \mu L_A \|\tilde{G}_{t-1}\|_F \leq \mu L_A \|P_{t-1}\|_F + \mu L_B L_C M^2 \|P_{t-1}\|_F + \mu L_B L_C M^2 \|P_{t-1}\|_F \\
&= \mu (L_A + 2L_B L_C M^2) \|P_{t-1}\|_F
\end{aligned} \quad (31)$$

To calculate the norm bound for the first term in (25), we first need to establish the bounds for \hat{D}_t . This involves estimating the minimum eigenvalue of \hat{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, \hat{D}_t will satisfy the following inequality for every unit vector \vec{v} :

$$\vec{v}^\top \hat{D}_t \vec{v} = \frac{1}{N} \sum_i \vec{v}^\top \left[(S^l{}^\top B_{i,t} S^l) \otimes (S^r{}^\top C_{i,t} S^r) \right] \vec{v} \geq \frac{1}{N} \sum_i \gamma_{min,i,t} \quad (32)$$

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

$$\lambda_{\max}(I - \mu \hat{D}_{t-1}) \leq 1 - \frac{\mu}{N} \sum_i \gamma_{min,i,t-1} \quad (33)$$

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

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

and completing the proof.

While SubTrack++ utilizes right/left projections to reduce memory consumption, the proof is presented using both projection matrices to ensure generality. Here, we demonstrate how the proof proceeds under the assumption $m \leq n$ (without loss of generality), which allows the use of the left projection matrix.

Using the left projection matrix, the current formulation of P_t , defined as $P_t = S_t^l{}^\top G_t S_t^r$, simplifies to $P_t = S_t^l{}^\top G_t$. Similarly, $\hat{G}_t = S_t^l S_t^l{}^\top G_t S_t^r S_t^r{}^\top$ reduces to $\hat{G}_t = S_t^l S_t^l{}^\top G_t$. From this point, the proof continues by substituting S_t^r with the identity matrix, allowing the derivation of the vectorized forms of g_t, \hat{g}_t, p_t , and related terms.

The remainder of the proof remains largely unaffected. It can be readily verified that the recursive formulation of g_t is unchanged. Although the definition of P_t is modified, it continues to satisfy the bounds required for convergence, ensuring that P_t converges to 0 when the left projection matrix is used.

B Grassmann Exponential

Theorem 3.6 (Grassmann Exponential). *Let $P = UU^\top \in Gr(n, p)$ be a point on the Grassmannian, where $U \in St(n, p)$ is the orthonormal basis of the corresponding subspace. Consider a tangent vector $\Delta \in T_P Gr(n, p)$, and let Δ_U^{hor} denote the horizontal lift of Δ to the horizontal space at U in the Stiefel manifold $St(n, p)$. Suppose the thin SVD of Δ_U^{hor} is given by $\Delta_U^{\text{hor}} = \hat{Q} \Sigma V^\top$, where $\hat{Q} \in St(n, r)$, $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$ contains the nonzero singular values of Δ_U^{hor} with $r = \min(p, n - p)$, and $V \in St(p, r)$. The Grassmann exponential map, representing the geodesic emanating from P in the direction Δ , is given by:*

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

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

$$\Delta = Q \begin{pmatrix} 0 & B^\top \\ B & 0 \end{pmatrix} Q^\top = \left[Q \begin{pmatrix} 0 & -B^\top \\ B & 0 \end{pmatrix} Q^\top, P \right] \quad (34)$$

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

$$\Delta_Q^{\text{hor}} = [\Delta, P]Q = Q \begin{pmatrix} 0 & -B^\top \\ B & 0 \end{pmatrix} \quad (35)$$

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

Definition B.1 (Orthogonal Group). The orthogonal group $O(n)$ is defined as the set of all $n \times n$ matrices Q over \mathbb{R} such that $Q^\top Q = QQ^\top = I_n$, where Q^\top is the transpose of Q and I_n is the $n \times n$ identity matrix:

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

Then the tangent space of the orthogonal group $O(n)$ at a point Q , denoted $T_Q O(n)$, 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., $\Omega^\top = -\Omega$:

$$T_Q O(n) = \{Q\Omega \mid \Omega \in \mathbb{R}^{n \times n}, \Omega^\top = -\Omega\}.$$

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

$$\text{Exp}_Q^O(tQ\Omega) = Q \exp_m(t\Omega), \quad (36)$$

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

$$\text{Exp}_P^{Gr}(t\Delta) = \pi^{OG} \left(Q \exp_m \left(t \begin{pmatrix} 0 & -B^\top \\ B & 0 \end{pmatrix} \right) \right). \quad (37)$$

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

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

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

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

which leads to the desired result when inserted into (37). For more mathematical details, you can refer to [Edelman et al. \[1998\]](#), [Bendokat et al. \[2024\]](#), or other useful resources on Grassmann geometry.

C Projection-Aware Optimizer

When projecting into a lower-dimensional space and tracking coordinate changes, we typically use orthonormal projection matrices to represent the subspaces and their multiplications to effect a change of basis. While this works well for purely linear operations, Adam's updates also incorporate non-linear elements.

Suppose the subspace changes at step t , transitioning from the subspace spanned by the orthonormal matrix S_{t-1} to that spanned by S_t . Since both matrices are orthonormal—preserved by the Grassmannian update rule in (5)—the matrix $S_t^\top S_{t-1}$ represents the change of basis between the two subspaces. In other words, if $\mathbb{E}_{t-1} = (e_{t-1}^1, \dots, e_{t-1}^r)$ and $\mathbb{E}_t = (e_t^1, \dots, e_t^r)$ are orthonormal bases for the subspaces at steps $t-1$ and t , respectively, then the i th column of matrix X transforms under the change of basis as $X_t^i = \sum_{j=1}^r \langle e_t^i, e_{t-1}^j \rangle X_{t-1}^j$, where X_{t-1}^j is the j th column of X based on the basis of the time step $t-1$.

$\mathbf{E}_{t,\beta}[\cdot]$ denotes the exponential time-weighted expectation at time t with decay rate β . Following the reinterpretation by [Robert et al. \[2025\]](#), Adam's first and second moment estimates can be expressed as $\tilde{M}_t = \mathbf{E}_{t,\beta_1}[\tilde{G}_t]$ and $\tilde{V}_t = \mathbf{E}_{t,\beta_2}[(\tilde{G}_t)^2]$, where \tilde{G}_t denotes the low-rank representation of the gradient at time step t . As shown in (38), the first moment estimate can be transformed under a change of basis using the change-of-basis matrix, $S_t^\top S_{t-1}$. Notably, $\langle \tilde{G}_t, e_t^i \rangle$ gives the i th column of \tilde{G}_t when the subspace has the basis \mathbb{E}_t . We use the superscripts to indicate a column of a matrix.

$$\mathbf{E}_{t,\beta_1}[\langle \tilde{G}_t, e_t^i \rangle] = \sum_{j=1}^r \langle e_t^i, e_{t-1}^j \rangle \mathbf{E}_{t,\beta_1}[\langle \tilde{G}_t, e_{t-1}^j \rangle] = \sum_{j=1}^r \langle e_t^i, e_{t-1}^j \rangle \tilde{M}_t^j = \left(S_t^\top S_{t-1} \tilde{M}_t \right)^i \quad (38)$$

Following the same approach, we can change the basis for the second moment estimate as described in (39).

$$\begin{aligned}
\mathbf{E}_{t,\beta_2} \left[\left(\langle \tilde{G}_t, e_t^i \rangle \right)^2 \right] &= \sum_{j=1}^r \langle e_t^i, e_{t-1}^j \rangle^2 \mathbf{E}_{t,\beta_2} \left[\left(\langle \tilde{G}_t, e_{t-1}^j \rangle \right)^2 \right] \\
&\quad + \sum_{k \neq l}^r \langle e_t^i, e_{t-1}^k \rangle \langle e_t^i, e_{t-1}^l \rangle \mathbf{E}_{t,\beta_2} \left[\langle \tilde{G}_t, e_{t-1}^k \rangle \langle \tilde{G}_t, e_{t-1}^l \rangle \right] \\
&= \sum_{j=1}^r \langle e_t^i, e_{t-1}^j \rangle^2 \tilde{\mathcal{V}}_t^j \\
&\quad + \sum_{k \neq l}^r \langle e_t^i, e_{t-1}^k \rangle \langle e_t^i, e_{t-1}^l \rangle \tilde{M}_t^k \tilde{M}_t^l.
\end{aligned} \tag{39}$$

In transitioning from the first equality to the second, we assume independence among the gradient coordinates. This enables us to approximate the covariance using a product of first-order moment estimates. This assumption is often reasonable in practice because we compute the SVD of the gradient and maintain an orthonormal subspace projection matrix, updating it along the Grassmannian geodesic to track the optimal subspace. Since SVD tends to diagonalize the covariance, the off-diagonal entries are typically negligible. Additionally, we clip any negative values to zero to ensure valid (non-negative) variance estimates. Moreover, to rewrite the second term in the final equality of (39), we employ the following equation:

$$\begin{aligned}
&\sum_k \sum_l \langle e_t^i, e_{t-1}^k \rangle \langle e_t^i, e_{t-1}^l \rangle \tilde{M}_t^k \tilde{M}_t^l \\
&= \sum_k \langle e_t^i, e_{t-1}^k \rangle^2 \left(\tilde{M}_t^k \right)^2 + \sum_{k \neq l} \langle e_t^i, e_{t-1}^k \rangle \langle e_t^i, e_{t-1}^l \rangle \tilde{M}_t^k \tilde{M}_t^l
\end{aligned} \tag{40}$$

Given these, we can rewrite (39) as follows:

$$\begin{aligned}
\mathbf{E}_{t,\beta_2} \left[\left(\langle \tilde{G}_t, e_t^i \rangle \right)^2 \right] &= \sum_j \langle e_t^i, e_{t-1}^j \rangle^2 \tilde{\mathcal{V}}_t^j + \\
&\quad \left[\sum_k \sum_l \langle e_t^i, e_{t-1}^k \rangle \langle e_t^i, e_{t-1}^l \rangle \tilde{M}_t^k \tilde{M}_t^l - \sum_k \langle e_t^i, e_{t-1}^k \rangle^2 \left(\tilde{M}_t^k \right)^2 \right] \\
&= \sum_j \langle e_t^i, e_{t-1}^j \rangle^2 \left[\tilde{\mathcal{V}}_t^j - \left(\tilde{M}_t^j \right)^2 \right] + \left(\langle e_t^i, e_{t-1}^j \rangle \tilde{M}_t^j \right)^2 \\
&= \left(\left(S_t^\top S_{t-1} \right)^2 \left[\tilde{\mathcal{V}}_t - \tilde{M}_t^2 \right] \right)^i + \left(\left(S_t^\top S_{t-1} \tilde{M}_t \right)^2 \right)^i
\end{aligned} \tag{41}$$

By applying (38) and (41), we can directly derive the update rules for the projection-aware optimizer, as expressed in (8) and (9).

D Time Complexity Analysis

Table 3 presents the time complexity breakdown for the subspace update step in the SubTrack++ algorithm assuming a $m \times n$ gradient matrix and rank r projection matrix, where $r \ll m \leq n$. As outlined in Algorithm 1, the subspace update step begins by solving the least squares problem (2) to estimate the optimal update for S_t , the $m \times r$ orthonormal matrix. This operation has a time complexity of $O(mr^2)$. Computing the residual and the partial derivative with respect to S_t requires $O(mrn)$ and $O(mnr)$ time respectively. This is because the solution to the least squares problem, A , has shape $r \times n$ which is multiplied by S_t in the residual $R = G_t - S_t A$, resulting in time complexity $O(mrn)$. The following operation for the partial derivative is $-2RA^\top$, where the matrix multiplication has $O(mnr)$ complexity. The tangent vector computation (4) which involves an identity transformation and matrix multiplication has time complexity of $O(m^2r)$.

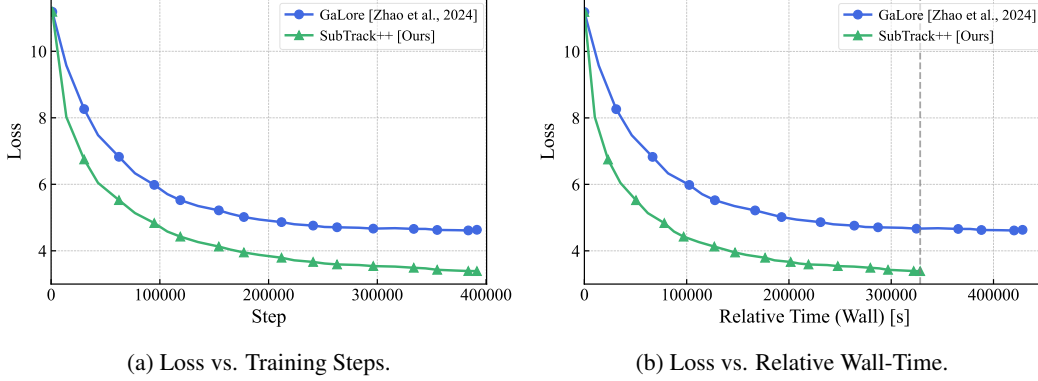


Figure 7: Comparison of pre-training Llama-7B architecture for 100k iterations. (a) shows training loss (\downarrow) versus training steps. (b) shows the same runs against wall-time. SubTrack++ outperforms GaLore; substantially reducing wall-time.

The rank-1 approximation step uses largest singular value from the SVD of the $m \times r$ tangent vector, and has time complexity of $O(mr^2)$. Finally, the update rule as shown in (13) which has a time complexity of $O(mr^2)$. The overall complexity of the algorithm is dominated by the matrix multiplication calculations of time complexity $O(mnr)$. However, unlike GaLore, since we avoid computing SVD operation on the $m \times n$ gradient matrix, which has complexity of $O(nm^2)$, the overall update step in SubTrack++ is still more efficient with respect to time complexity.

Table 3: Time Complexity for SubTrack++ Subspace Update

Computation Step	Time
Cost function	$O(mr^2)$
Residual	$O(mrn)$
Partial derivative	$O(mnr)$
Tangent vector ΔF	$O(m^2r)$
Rank-1 approximation of ΔF	$O(mr^2)$
Update rule	$O(mr^2)$
Overall	$O(mnr)$

E Pre-Training Llama-Based Architectures

We pre-trained all six Llama-based architectures for 10k iterations using hyperparameters reported in Table 4. To demonstrate the generalizability of the proposed method, we also present results from pre-training the 7B architecture with both SubTrack++ and GaLore for 100k iterations, as shown in Figure 7. SubTrack++ maintains its advantage in terms of faster convergence and superior performance. The hyperparameters of this run are identical to those reported in Table 4, except for the number of iterations which is 100k. Also the bar-chart version of Figure 1 is represented in Figure 8.

F Memory and Time Comparison

Table 5 presents the the peak memory consumption measured to compare SubTrack++ and other baselines. It shows that SubTrack++ requires on-par or better memory compared to GaLore [Zhao et al., 2024] while getting state-of-the-art results. As detailed in Table 2, all baselines except BAdam [Luo et al., 2024] use the same number of optimizer parameters; therefore, any differences in their peak memory consumption stem from variations in their runtime parameters.

Additionally, Table 6 presents the wall-time consumed by each baseline across all model architectures during pre-training. Each run is configured to include exactly 10 subspace updates for SubTrack++ and other baselines employing periodic subspace updates. Specifically, models ranging from 60M to 3B use a subspace update interval of 200, resulting in 2,000 total iterations, while the 7B models use an interval of 500, yielding 5,000 iterations. Experiments for the 60M to 3B models are conducted on an NVIDIA A100 GPU, while the 7B model experiments are run on an NVIDIA RTX A6000.

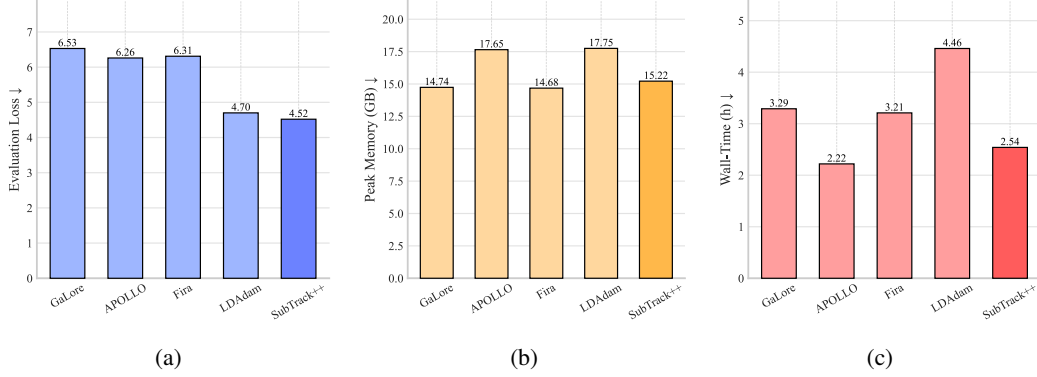


Figure 8: We compare baselines on pre-training a 1B-parameter model. (a) SubTrack++ achieves the lowest evaluation loss across all methods. (b) Its peak memory usage is significantly lower than APOLLO and LDAdam, and on par with GaLore and Fira. (c) In terms of wall-time, SubTrack++ incurs minimal overhead relative to APOLLO and is markedly faster than GaLore, Fira, and LDAdam. Overall, SubTrack++ outperforms all baselines in evaluation loss while matching or exceeding them in memory and runtime efficiency.

Table 4: Hyperparameters of pre-training Llama-based architectures.

		60M	130M	350M	1B	3B	7B
Architectural Parameters	Hidden	512	768	1024	2048	2560	4096
	Intermediate	1376	2048	2736	5461	6848	11008
	Heads	8	12	16	24	32	32
	Layers	8	12	24	32	32	32
Shared Parameters	Learning Rate	1e-3	1e-3	1e-3	1e-4	1e-4	1e-4
	Batch Size	128	128	64	8	8	4
	Gradient Accumulation	2	2	2	2	2	4
	Iterations				10k		
	Gradient Clipping				1.0		
	Warmup Steps				1000		
	scale dtype				0.25 bfloat16		
Low-Rank Optimizer Methods Parameters	Rank	128	256	256	512	512	1024
	Subspace Update Interval	200	200	200	200	200	500
	SubTrack++ Step-Size				10000		
BAdam Parameters	Block Switch Interval				100		
	Switch Mode				Random		

G Fine-Tuning Experiments

As described, we examined SubTrack++ on fine-tuning RoBERTa-base and RoBERTa-large models to evaluate them on GLUE and SuperGLUE benchmarks. The results for GLUE task is summarized in Table 7, and SuperGLUE in 8.

The hyperparameters for fine-tuning RoBERTa-base are detailed in Table 10, matching those reported in the GaLore [Zhao et al., 2024] for rank-8 subspaces, with a subspace update interval set at 500 iterations. We also fine-tuned RoBERTa-Large on SuperGLUE tasks using the hyperparameters from Luo et al. [2024], as detailed in Table 11, with the exception that we fine-tuned each task for 30 epochs.

The results of supervised fine-tuning of the Llama-2-7B-chat-hf model on the Alpaca dataset on an Nvidia-H100 GPU are presented in Table 9. The fine-tuning was performed for one epoch, and the corresponding hyperparameters are listed in Table 12.

Table 5: Peak memory consumption of pre-training Llama-based architectures on C4 dataset. The 7B models are trained using the 8-bit Adam optimizer, except for the runs marked with *. SubTrack++ demonstrates better or on-par memory compared to other low-rank methods that allows full-parameter training.

	60M r=128	130M r=256	350M r=256	1B r=512	3B r=512	7B r=1024
Full-Rank	16.86	25.32	28.67	18.83	34.92	50.50
BAdam [Luo et al., 2024]	13.34	20.01	16.45	9.18	14.75	22.35
GaLore [Zhao et al., 2024]	16.89	25.52	27.85	14.74	26.03	36.00
Online Subspace Descent [Liang et al., 2024]	16.61	25.86	28.76	18.45	32.57	33.10
LDAdam [Robert et al., 2025]	17.18	25.94	28.03	18.45	32.57	OOM*
Fira [Chen et al., 2025]	16.39	24.99	27.33	14.68	25.62	47.84*
SubTrack++ (Ours)	16.40	25.06	27.42	15.22	25.54	49.82*

Table 6: Wall-time comparison of pre-training Llama-based architectures on the C4 dataset. The number of iterations is set to ensure 10 subspace updates for methods using periodic subspace adjustments. SubTrack++ achieves the lowest wall-time among all baselines that support full-parameter training on large models. The 7B models are trained using the 8-bit Adam optimizer, except for the runs marked with *. Since this can impact wall-time comparisons, it is more appropriate to compare runs within the same cluster.

	60M r=128	130M r=256	350M r=256	1B r=512	3B r=512	7B r=1024
Full-Rank	524.0	1035.1	1396.4	974.9	1055.9	12726.2
BAdam [Luo et al., 2024]	511.3	779.2	961.6	798.6	1004.1	7283.7
GaLore [Zhao et al., 2024]	547.8	1094.2	1589.0	1729.5	2715.5	21590.4
Online Subspace Descent [Liang et al., 2024]	662.8	1228.2	1818.6	1438.7	1676.9	21590.4
LDAdam [Robert et al., 2025]	639.9	1342.2	2083.4	2780.9	4625.4	OOM*
Fira [Chen et al., 2025]	635.3	1180.8	1729.7	1938.5	2898.4	22554.3*
SubTrack++ (Ours)	627.6	1140.6	1593.2	1304.3	1517.6	16491.7*

Table 7: Evaluating the performance of SubTrack++ and other baselines when fine-tuning RoBERTa-Base on GLUE tasks for $r = 8$. The performance is measured via Accuracy (\uparrow) for SST-2 and RTE tasks, F1 (\uparrow) for MRPC, Pearson Correlation (\uparrow) for STS-B, and Matthews Correlation (\uparrow) for COLA. The best results are marked in **bold**, with the second-best performance underlined.

	COLA	STS-B	MRPC	RTE	SST-2
Full-Rank	62.57	91.03	91.32	77.98	94.27
BAdam [Luo et al., 2024]	54.44	89.01	91.35	68.59	94.15
GaLore [Zhao et al., 2024]	58.54	90.61	91.30	74.37	94.50
LDAdam [Robert et al., 2025]	58.81	<u>90.90</u>	92.22	<u>76.53</u>	<u>94.27</u>
SubTrack++ (Ours)	<u>58.55</u>	90.95	<u>92.04</u>	78.34	90.02

Table 8: Evaluating the performance of SubTrack++ and other baselines when fine-tuning RoBERTa-Large on SuperGLUE tasks with $r = 8$. The performance is measured via Accuracy (\uparrow) for COPA, WIC, WSC, BoolQ, and AX_g tasks, and F1 (\uparrow) for CB. The best results are marked in **bold**, with the second-best performance underlined.

	BoolQ	CB	COPA	WIC	WSC	AX _g
Full-Rank	85.96	90.33	76.00	71.79	63.46	96.30
GaLore [Zhao et al., 2024]	<u>85.44</u>	88.85	<u>80.00</u>	71.47	<u>63.46</u>	100.00
BAdam [Luo et al., 2024]	82.51	53.28	59.00	70.38	60.58	51.85
LDAdam [Robert et al., 2025]	85.75	56.16	<u>80.00</u>	<u>71.00</u>	64.42	<u>70.37</u>
SubTrack++ (Ours)	85.38	<u>83.96</u>	82.00	70.70	62.5	100.00

Table 9: Comparing final evaluation perplexity (\downarrow) and wall-time of fine-tuning Llama-2-7B-chat-hf on Alpaca dataset with GaLore and SubTrack++ .

Method	Evaluation Perplexity	Wall-Time (min)
GaLore	2.43	110.8
SubTrack++	2.41	73.2

Table 10: Hyperparameters of fine-tuning RoBERTa-Base on GLUE tasks.

		SST-2	MRPC	CoLA	RTE	STS-B
Shared Parameters	Batch Size # Epochs Max Seq. Len.	16	16	32 30 512	16	16
Low-Rank Optimizer Methods Parameters	Learning Rate SubTrack Step-Size Subspace Update Interval Rank Config α	2E-05 0.1	2E-05 3.0	1E-05 5.0 500 8 2	2E-05 15.0	3E-05 10.0
BAdam Parameters	Learning Rate Block Switch Interval Switch Mode	2E-05	2E-05	1E-05 100 Random	2E-05	3E-05

Table 11: Hyperparameters of fine-tuning RoBERTa-Large on SuperGLUE tasks.

		BoolQ	CB	COPA	WIC	WSC	AX _g
Shared Parameters	Batch Size # Epochs Learning Rate Max Seq. Len.				16 30 1e-5 512		
Low-Rank Optimizer Methods Parameters	SubTrack++ Step-Size Subspace Update Interval Rank Config. α	0.1 500	10.0 100	10.0 100 8 4	100.0 500	1.0 250	1.0 100
BAdam Parameters	Block Switch Interval Switch Mode	100	50	50 Random	100	50	50

Table 12: Hyperparameters of fine-tuning Llama-2-7B-chat-hf on Alpaca dataset.

Parameter	Value
Subspace Update Interval	500
Rank	1024
α	0.25
Target Modules	att, mlp
Batch Size	8
Epoch	1