Stabilizing the Dynamic Low-Rank Training

Zhonghan Xu Ling Wang Junhao Chen Jianwei Zhao Jinwei Yang
School of Communications and Information Engineering
University of Electronic Science and Technology of China

Abstract

Training neural networks directly in a low-rank parameterization is an appealing route to reducing memory, compute, and storage simultaneously during both training and inference. Dynamic low-rank training (DLRT), which confines weights to a rank-r manifold via the Galerkin projection of the gradient flow, is particularly attractive because it identifies efficient subnetworks on the fly without specialized initialization or post-factorization. However, DLRT fails to find trainable networks under high compression. In this paper, we derive the gradient flow of the best rank-r approximation and point out that the offset of DLRT comes from a curvature-coupling term which is large and thus non-negligible under aggressive compression. Guided by this analysis, we propose a stable dynamic low-rank training method, named SDLRT, which maintains a lightweight compensation buffer that reinjects the top neglected singular directions. Additionally, we introduce a negative feedback on the truncation tolerance to stabilize each layer's rank. Experimentally, SDLRT reliably finds trainable subnetworks where DLRT collapses and as a PEFT adapter on DeBERTa-v3, it achieves the best average score on SuperGLUE at only 2.8% parameter overhead over LoRA.

1 Introduction

Modern neural networks deliver state-of-the-art performance across a wide range of applications, but their ever-growing parameter counts make deployment on memory- and compute-constrained devices increasingly difficult. A natural remedy is to exploit the empirical observation that the weight matrices of over-parameterized networks are approximately low-rank, and to train the network directly in a low-rank parameterization so that memory, compute, and storage are reduced simultaneously in both the training and inference phases [1].

Among existing low-rank training methods, dynamic low-rank training (DLRT) proposed by [2] is particularly attractive: it constrains each weight matrix to a low-rank manifold via the Galerkin condition and updates only the three Singular Value Decomposition (SVD) factors (U, S, V), thereby avoiding specialized initialization and the costly post-factorization step. Unlike iterative magnitude pruning [3] or Neural Architecture Search-based compression pipelines [4], DLRT identifies low-rank "winning tickets" on the fly, with resource usage scaling linearly in the current rank rather than in the ambient dimension.

However, we observe that DLRT becomes unreliable precisely where compression matters most. As illustrated in Figure 1, on VGG-19/CIFAR-100 with an about 69.6% initial compression, DLRT fails to find a trainable subnetwork: ranks contract rapidly in the early epochs and the truncated singular-value tail continues to carry the majority of each layer's energy throughout training. This echoes the expansion-limitation phenomenon reported by [5], and here it takes the form of a self-reinforcing loop in which reduced expressive capacity drives ranks downward, and lower ranks further erode expressive capacity.

Fortunately, some scholars have pointed out that whether a trainable subnetwork can be found is related to the stability of the sub-network [6]. We argue that the low-rank instability is not a mere

Preprint.

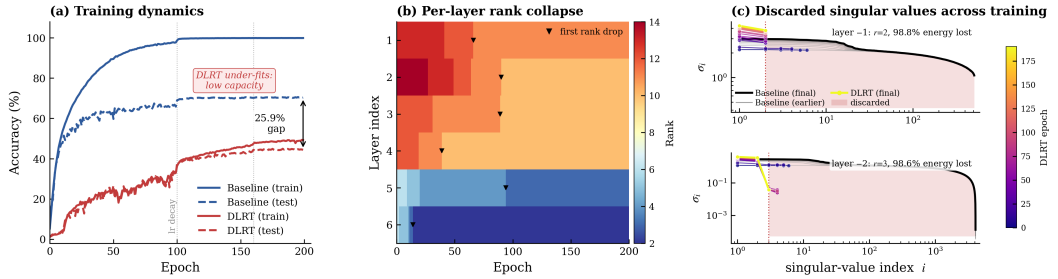


Figure 1: **DLRT collapses under aggressive compression on VGG-19/CIFAR-100 ($\approx 69.6\%$ compression).** (a) Accuracy curves: DLRT lags the baseline by 25.9% in test accuracy; training accuracy also plateaus far below baseline, indicating insufficient capacity rather than over-fitting. (b) Per-layer rank evolution; ▼ marks the first epoch each layer hits its minimum rank—every layer collapses within the first 100 epochs. (c) Singular-value spectra of the last two weight matrices across epochs $\{0, 10, \ldots, 190\}$. The red shaded tail beyond $i > r$ is the energy that DLRT permanently discards: $98.8\%/98.6\%$ of the total $\|W\|_F^2$ for the two layers.

optimization artifact but a direct consequence of the gradient flow based on dynamic low-rank approximation (DLRA) [7]. Inspired by the analysis of dynamic orthogonal approximation in [7], we derive the gradient flow satisfied by the best rank-r approximation of the full-parameter weights and compare it with DLRA. The comparison reveals two distinct sources of error: *gradient locality*—DLRA evaluates the gradient at the low-rank neighbour rather than at the full-parameter point, which is a price we must pay for staying on the manifold; and *curvature coupling*—DLRA entirely drops a correction term in which the discarded singular components $(\sigma_{r+j}, u_{r+j}, v_{r+j})$ interact with the retained ones (σ_i, u_i, v_i) through a factor proportional to $\sigma_{r+j}/(\sigma_i^2 - \sigma_{r+j}^2)$. Under aggressive compression the spectral gap $\sigma_i^2 - \sigma_{r+j}^2$ is small, so the coupling is large and its omission is the dominant source of the instability observed in Figure 1.

Guided by this analysis, we propose **Stabilized Dynamic Low-Rank Training (SDLRT)**. Rather than raising the rank, which is expensive, SDLRT keeps a lightweight *compensation buffer* $(U_{\text{neg}}, V_{\text{neg}})$ that stores the top neglected singular directions from the previous iteration and reinjects them when updating the small factor S, partially recovering the curvature-coupling term missing from DLRT. A negative feedback mechanism on the singular-value tolerance ϑ further stabilizes the rank of each layer. Our main contributions are summarized as follows:

1. We derive the gradient flow of the best rank-r approximation of the full-parameter weights in closed SVD form (Theorem 3.1) and identify the curvature-coupling term whose omission makes DLRA unstable under high compression.
2. Built on this insight, we propose a stabilized dynamic low-rank training method, which augments the basis-extension step with the compensation buffer and stabilizes the adapted rank through a negative feedback on the truncation tolerance ϑ. Additionally, we provide an exactness guarantee (Theorem 3.2) and an error bound (Theorem 3.3), confirming that the buffer does not alter the starting point and that convergence is preserved.
3. Extensive experiments across computer vision tasks and natural language processing show that SDLRT reliably identifies trainable low-rank subnetworks under aggressive compression, and significantly improves performance on natural language understanding.

2 Related Work

Model compression. Pruning is the dominant route to reducing the memory and compute footprint of deep networks. Unstructured pruning attains high sparsity but yields limited speedup without specialized kernels [8–10], whereas structured pruning removes entire channels, filters, or attention heads at the cost of sharper accuracy degradation under aggressive compression [11–14]. A more fundamental bottleneck is the *pretrain–prune–retrain* pipeline itself: its iterative cost scales poorly with model size, and the final retraining step often fails to recover accuracy once the compression ratio exceeds a task-dependent threshold [15–17]. Low-rank factorization offers a complementary

view by pruning redundant directions rather than individual entries [18–20]. Yet applying post-SVD still requires a fully trained dense model and decouples the rank decision.

Lottery ticket hypothesis. Some scholars conjectured that dense networks contain sparse "winning tickets" which, trained in isolation, match the accuracy of the full model [3]. Subsequent work stabilized the hypothesis at scale through weight and learning-rate rewinding [16, 5], established transferability across datasets and tasks [21–23], and proved strong-lottery variants in which subnetworks are uncovered in randomly initialized networks without any weight update [24–27]. The hypothesis has since been extended to GNNs [28, 29], GANs and diffusion models [30, 31], and pretrained language models [23, 32]. Nevertheless, identifying a winning ticket remains the central difficulty: iterative magnitude pruning demands many train–prune cycles [33, 34], and sanity checks reveal that the outcome is highly sensitive to learning rate, training length, and residual-connection topology [35, 36]. A parallel line pursues low-rank winning tickets in spectral space [2]; however, as we show in Section 3.2, the standard dynamic low-rank procedure used to uncover them collapses precisely when compression matters most.

Dynamic low-rank approximation. Dynamic low-rank approximation (DLRA) restricts optimization to the manifold of rank-r matrices via a Galerkin projection of the gradient flow [7], with robust numerical integrators developed in [37, 38]. Building on these tools, DLRT [2] and its rank-adaptive extension [39] update only the SVD factors (U, S, V) during training, eliminating both specialized initialization and post-hoc factorization. Despite this progress, a shared weakness persists: at every step these methods *discard the truncated spectral tail*, implicitly treating small singular components as irrelevant. In Section 3.2 we show that this omission drops a curvature-coupling term under aggressive compression and drives the rank collapse as shown in Figure 1. Our method directly targets this neglected interaction via a lightweight compensation buffer, at negligible additional cost.

3 Method

3.1 Low-rank Training

Since over-parameterized models tend to have low-rank property [40, 41], low-rank methods are commonly used strategies to reduce memory storage and computational costs during the training and inference phases of deep learning models. As shown in Figure 2, low-rank training based on SVD [42, 43] parameterizes the weights of each layer W as $W = USV^\top$, where U and V are orthogonal matrices with r columns, and S is a small matrix of size $r \times r$. During model training or inference, the update of the weight matrix is transformed into updates of the low-rank factors U, V, S, which can significantly save resources. However, unconstrained alternating or direct training may cause large oscillations in the training loss and slower convergence [44, 45]. It is known that the maximum curvature (after normalizing the normal vector) of the point on the low-rank manifold corresponds to the inverse of the smallest singular value [46]. Therefore, when the smallest singular value approaches zero, the curvature at that point diverges. To ensure the accuracy of the approximation, the largest singular value that is ignored must be very small, meaning that the remained smallest singular value should not be too large. To mitigate this phenomenon, most researchers address it from the aspect: the constraint of the training process [47, 2, 48].

Given the loss function $\mathcal{L}$, we assume that the training phase is considered as a continuous process that changes over time, where the weights of the full-parameter network are denoted as $W_f(t)$, and their time-derivative is:

$$W_f(t) = W_f(0) + \int_0^t \dot{W}_f(s)\, ds, \quad \text{where} \quad \dot{W}_f(s) = -\nabla_{W_f} \mathcal{L}(W_f(s)). \tag{1}$$

To constrain the full-parameter weights onto a low-rank manifold, DLRA projects the initial point onto a rank-r manifold and subsequently projects the gradients onto the tangent plane of the manifold:

$$\dot{W}(t) = \Pi_{\mathcal{T}(W)}(-\nabla_W \mathcal{L}(W(t))) \quad \text{and} \quad W(0) = \Pi_{\mathcal{M}_r}(W_f(0)), \tag{2}$$

where $W(t)$ denotes the low-rank weights projected by DLRA, $\mathcal{M}_r$ denotes the manifold of rank-r matrices, $\mathcal{T}(W)$ is the tangent space at $W \in \mathcal{M}_r$, $\Pi_{\mathcal{T}(W)}$ is the orthogonal projection onto the tangent space $\mathcal{T}(W)$, and $\Pi_{\mathcal{M}_r}$ is the orthogonal projection onto the manifold $\mathcal{M}_r$, i.e., the map that keeps the first r singular values and sets the rest to zero.

Specifically, assuming the low-rank network weights are parameterized as $W(t) = USV^\top \in \mathcal{M}_r$, the optimization problem we consider is formulated as follows:

$$\begin{aligned} \min \quad & \|\dot{W}(t) + \nabla_W \mathcal{L}(W(t))\|_F \\ \text{s.t.} \quad & W(t) = USV^\top \in \mathcal{M}_r \\ & \dot{W}(t) = \dot{U}SV^\top + U\dot{S}V^\top + US\dot{V}^\top \in \mathcal{T}(W(t)). \end{aligned} \tag{3}$$

Then, using the Gauge conditions $U^\top \dot{U} = 0$ and $V^\top \dot{V} = 0$ as presented in [7], we obtain the gradient flow equation for the low-rank factors:

$$\begin{cases} \dot{S} = -U^\top \nabla_W \mathcal{L}(W(t))V, \\ \dot{U} = -P_U^\perp \nabla_W \mathcal{L}(W(t))VS^{-1}, \\ \dot{V} = -P_V^\perp \nabla_W \mathcal{L}(W(t))^\top US^{-\top}, \end{cases} \tag{4}$$

where $P_U^\perp = (I - UU^\top)$ and $P_V^\perp = (I - VV^\top)$ are used to project the original matrix onto the spaces orthogonal to the span of U and V, respectively.

However, we observe that at higher compression rates, the low-rank networks trained based on Equation (4) are highly unstable as shown in Figure 1, with potentially low expressive power, making it difficult to identify trainable "winning tickets" in the network. In Figure 1, we plot the performance of VGG-19 trained with the DLRT method compared to the baseline under an initial compression rate of 69.6%, as well as the rank variations across multiple layers within the network. The instability of the network leads to unstable ranks of the network weights trained based on Equation (4). This rank instability, in turn, negatively impacts the expressive power of the network, further exacerbating its instability and creating a vicious cycle.

In fact, the weights obtained through training with Equation (4) still exhibit certain offset compared to the best low-rank approximation of the original weights derived from full-parameter gradient training.

3.2 Gradient Flow of the Best Low-Rank Approximation

To directly characterize the gap between DLRA and the best low-rank trajectory, we now derive the gradient flow equations satisfied by the best rank-r approximation of the full-parameter weight $W_f(t)$. The derivation makes explicit which information is discarded by DLRA and motivates the compensation mechanism introduced in Section 3.3.

Throughout this subsection, we write $\mathcal{G} \triangleq \nabla_W \mathcal{L}(W_f(t))$ for brevity and recall that $\dot{W}_f = -\mathcal{G}$.

Setup and notation. Let the SVD of the full-rank weight matrix be $W_f = \sum_{i=1}^{r+k} \sigma_i u_i v_i^\top$, partitioned as

$$W_f = U\,\Sigma_r V^\top + U_\perp \Sigma_\perp V_\perp^\top, \tag{5}$$

where $U = [u_1, \ldots, u_r]$, $U_\perp = [u_{r+1}, \ldots, u_{r+k}]$, $V = [v_1, \ldots, v_r]$, $V_\perp = [v_{r+1}, \ldots, v_{r+k}]$, $\Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, and $\Sigma_\perp = \mathrm{diag}(\sigma_{r+1}, \ldots, \sigma_{r+k})$. The best rank-$r$ approximation is $USV^\top$ with $S = \Sigma_r$. We adopt the gauge conditions $U^\top \dot{U} = 0$ and $V^\top \dot{V} = 0$, which guarantee that $U^\top U = I_r$ and $V^\top V = I_r$ are preserved along the flow. The following orthogonality relations will be used repeatedly:

$$W_f V = U\Sigma_r, \quad W_f^\top U = V\Sigma_r, \quad V_\perp^\top V = 0, \quad U_\perp^\top U = 0. \tag{6}$$

Deriving the coupled system. The optimality of the rank-r truncation requires $(I - UU^\top)W_f V = 0$ and $(I - VV^\top)W_f^\top U = 0$ at all times. Differentiating the first constraint and using (6):

$$-\dot{U}\underbrace{U^\top W_f V}_{=\Sigma_r} - U\underbrace{\dot{U}^\top W_f V}_{=\dot{U}^\top U\Sigma_r = 0} + P_U^\perp \dot{W}_f V + P_U^\perp W_f \dot{V} = 0, \tag{7}$$

where $P_U^\perp = I - UU^\top$. Since $P_U^\perp W_f (I - VV^\top) = U_\perp \Sigma_\perp V_\perp^\top$ and $\dot{V} = (I - VV^\top)\dot{V}$ by the gauge conditions, we obtain

$$\dot{U}\Sigma_r = -P_U^\perp \mathcal{G} V + U_\perp \Sigma_\perp V_\perp^\top \dot{V}. \tag{8}$$

By symmetry, differentiating $(I - VV^\top)W_f^\top U = 0$ yields

$$\dot{V}\Sigma_r = -P_V^\perp \mathcal{G}^\top U + V_\perp \Sigma_\perp U_\perp^\top \dot{U}. \tag{9}$$

Equations (8) and (9) are coupled through the off-diagonal block $\Sigma_\perp$, which encodes the very interaction between retained and discarded components that DLRA neglects.

Expansion in the singular-vector basis. By the gauge conditions, $\dot{U} \in \mathrm{col}(U)^\perp$ and $\dot{V} \in \mathrm{col}(V)^\perp$. We may therefore expand $\dot{U} = U_\perp \mathcal{C}$ and $\dot{V} = V_\perp \mathcal{D}$, with coefficient matrices $\mathcal{C}, \mathcal{D} \in \mathbb{R}^{k\times r}$. Substituting into (8) and multiplying to the left by $U_\perp^\top$ gives

$$\mathcal{C}\Sigma_r = -U_\perp^\top \mathcal{G} V + \Sigma_\perp \mathcal{D}, \tag{10}$$

while (9) analogously yields

$$\mathcal{D}\Sigma_r = -V_\perp^\top \mathcal{G}^\top U + \Sigma_\perp \mathcal{C}. \tag{11}$$

For each pair (i, j) with $1 \le i \le r$, $1 \le j \le k$, define the shorthand

$$a_{ji} \triangleq u_{r+j}^\top \mathcal{G} v_i, \qquad b_{ji} \triangleq v_{r+j}^\top \mathcal{G}^\top u_i = u_i^\top \mathcal{G} v_{r+j}. \tag{12}$$

Equations (10) and (11) read component-wise:

$$\sigma_i \mathcal{C}_{ji} = -a_{ji} + \sigma_{r+j}\mathcal{D}_{ji}, \qquad \sigma_i \mathcal{D}_{ji} = -b_{ji} + \sigma_{r+j}\mathcal{C}_{ji}. \tag{13}$$

Eliminating $\mathcal{D}_{ji}$ and using the distinct-singular-value assumption $\sigma_i^2 \neq \sigma_{r+j}^2$:

$$\begin{cases} \mathcal{C}_{ji} = -\dfrac{\sigma_i a_{ji} + \sigma_{r+j} b_{ji}}{\sigma_i^2 - \sigma_{r+j}^2}, \\ \mathcal{D}_{ji} = -\dfrac{\sigma_{r+j} a_{ji} + \sigma_i b_{ji}}{\sigma_i^2 - \sigma_{r+j}^2}. \end{cases} \tag{14}$$

Reassembling the evolution equations. Substituting (14) back into $\dot{U} = U_\perp \mathcal{C}$ and expressing the result as a rank-one sum:

$$\dot{U} = -P_U^\perp \mathcal{G} V S^{-1} - \left[\sum_{\substack{1\le i\le r \\ 1\le j\le k}} \frac{\sigma_{r+j}}{\sigma_i^2 - \sigma_{r+j}^2} (\sigma_{r+j} u_{r+j}^\top \mathcal{G} v_i + \sigma_i u_i^\top \mathcal{G} v_{r+j}) u_{r+j} v_i^\top \right] V S^{-1}, \tag{15}$$

where the first term arises from extracting the $\sigma_i^{-1} a_{ji}$ part of $\mathcal{C}_{ji}$, and the bracketed sum collects the remaining σ_{r+j}-dependent corrections. To verify the splitting, observe

$$\mathcal{C}_{ji} = \underbrace{-\frac{a_{ji}}{\sigma_i}}_{\text{DLRA part}} \underbrace{- \frac{\sigma_{r+j}(\sigma_{r+j} a_{ji} + \sigma_i b_{ji})}{\sigma_i(\sigma_i^2 - \sigma_{r+j}^2)}}_{\text{curvature correction}}, \tag{16}$$

which follows from the identity $\frac{\sigma_i}{\sigma_i^2 - \sigma_{r+j}^2} = \frac{1}{\sigma_i} + \frac{\sigma_{r+j}^2}{\sigma_i(\sigma_i^2 - \sigma_{r+j}^2)}$. The DLRA part equals $-U_\perp U_\perp^\top \mathcal{G}\, V \Sigma_r^{-1} = -P_U^\perp \mathcal{G}\, V S^{-1}$ since $P_U^\perp = U_\perp U_\perp^\top$ on $\mathrm{col}(U)^\perp$.

The equation for $\dot{V}$ follows by applying the same argument to (14) with the symmetrical exchange of the roles of (U, V) and (σ_i, σ_{r+j}).
Finally, the evolution of S is straightforward:

$$\dot{S} = \frac{d}{dt}(U^\top W_f V) = U^\top \dot{W}_f V = -U^\top \mathcal{G} V. \tag{17}$$

Collecting these three results yields exactly the coupled system stated in Theorem 3.1 below.

Theorem 3.1. *Let $W_f(t)$ denote the original network weights obtained through full-parameter gradient training, as defined in Equation* (1)*. Assume that there is no repeated singular value crossing*

the truncation, and let $USV^\top$ represent the best rank-r approximation of $W_f(t)$. Then its derivative is given by

$$\begin{cases} \dot{U} = -P_U^\perp \mathcal{G} V S^{-1} - \left[\sum_{\substack{1\le i\le r \\ 1\le j\le k}} (\mathcal{P}_{ij} u_{r+j}^\top \mathcal{G} v_i + \mathcal{Q}_{ij} u_i^\top \mathcal{G} v_{r+j}) u_{r+j} v_i^\top \right] V S^{-1}, \\ \dot{V} = -P_V^\perp \mathcal{G}^\top U S^{-T} - \left[\sum_{\substack{1\le i\le r \\ 1\le j\le k}} (\mathcal{P}_{ij} v_{r+j}^\top \mathcal{G}^\top u_i + \mathcal{Q}_{ij} v_i^\top \mathcal{G}^\top u_{r+j}) v_{r+j} u_i^\top \right] U S^{-T}, \\ \dot{S} = -U^\top \mathcal{G} V, \end{cases} \tag{18}$$

where $\mathcal{G} = \nabla_W \mathcal{L}(W_f(t))$, $\mathcal{P}_{ij} = \sigma_{r+j}^2/(\sigma_i^2 - \sigma_{r+j}^2)$, and $\mathcal{Q}_{ij} = \sigma_i \sigma_{r+j}/(\sigma_i^2 - \sigma_{r+j}^2)$.

Interpretation. Comparing Theorem 3.1 with the DLRA update in Equation (4), two discrepancies stand out:

(1) Gradient locality. The best rank-r flow evaluates the gradient $\nabla_W \mathcal{L}(W_f(t))$ at the full-parameter point $W_f(t)$, whereas DLRA evaluates the gradient $\nabla_W \mathcal{L}(W(t))$ at the neighboring low-rank point $W(t)$. This mismatch is a necessary price of working on the low-rank manifold and cannot be eliminated without sacrificing the memory savings.

(2) Curvature coupling. The bracketed correction terms, governed by the factor $\sigma_{r+j}/(\sigma_i^2 - \sigma_{r+j}^2)$, explicitly couple the retained singular triplets (σ_i, u_i, v_i) with the truncated ones $(\sigma_{r+j}, u_{r+j}, v_{r+j})$. When the gap $\sigma_i^2 - \sigma_{r+j}^2$ is small—precisely the regime encountered under aggressive compression—these corrections become non-negligible. DLRA drops them entirely, which we identify as the root cause of the rank-collapse and accuracy-degradation phenomena documented in Figure 1.

This analysis directly motivates our design: rather than discarding the truncated basis $(U_\perp, V_\perp)$ at each step, SDLRT *preserves it as a compensation buffer* $(U_{\text{neg}}, V_{\text{neg}})$ *and reinjects it into the subsequent update of* S, thereby partially recovering the curvature coupling missing from Equation (4).

3.3 Stabilizing the Dynamic Low-Rank Training

To better approximate the best rank-r approximation, we retain the discarded singular value components at each update to assist the next one, as shown in the right panel of Figure 2. Instead of increasing the rank r, which is computationally expensive, we only incorporate small singular value information when updating the small matrix S with the gradient flow, leaving the dimensions of the basis matrices unchanged. In this way, the additional computational and storage costs are basically only related to the size of the small matrix S. Since $r \ll \min(m, n)$, the additional costs introduced by this framework are essentially negligible. Based on the framework illustrated in the right panel of Figure 2, we propose a more stable variant of DLRT with the following update strategy for each layer's low-rank factors:

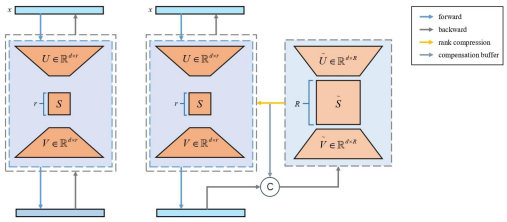


Figure 2: The procedures of SVD-based low-rank training (left) and the stable SVD-based low-rank training with compensation (right).

1. Initialize $K_0 = US$, $L_0 = VS^\top$ and update K and L separately via gradient flow to obtain K_1 and L_1.
2. Extend (K_1, U) and (L_1, V) with the truncated basis matrices U_{neg} and V_{neg} from the previous iteration to form the compensated matrices (K_1, U, U_{neg}) and (L_1, V, V_{neg}).
3. Project the compensated matrices onto the Stiefel manifold (e.g., via QR decomposition) to obtain the orthogonal bases U_1 and V_1.
4. Initialize $S_0 = U_1^\top W_{\text{prev}} V_1$, where $W_{\text{prev}} = USV^\top$. And in the newly updated space of matrices $U_1 S V_1^\top$, perform a Galerkin-based update [37] to obtain a new small matrix S_1.

5. Compute the SVD of S_1, i.e., $S_1 = \hat{U}\hat{S}\hat{V}^\top$, and mark the rank of S_1 as $r_{\max}$. Determine the new rank r_1 based on the tolerance $\vartheta = \tau\|\hat{S}\|$ and update the matrices:

$$\begin{cases} U_1 = U_1\hat{U}[:, : r_1],\ S_1 = \hat{S}[: r_1, : r_1],\ V_1 = V_1\hat{V}[:, : r_1], \\ U_{\text{neg}} = U_1\hat{U}[:, r_1 : \min(2r_1, r_{\max})],\ V_{\text{neg}} = V_1\hat{V}[:, r_1 : \min(2r_1, r_{\max})]. \end{cases}$$

6. Negative feedback mechanism: If the new rank r_1 is less than the initial rank r_0, multiply the tolerance factor τ by ω, where $\omega < 1$ is a feedback factor.

Connection to Theorem 3.1. The compensation is exactly the term Theorem 3.1 identifies as missing from DLRA. When the Galerkin update of Step 4 is carried out in the compensated basis containing $U_{\text{neg}}, V_{\text{neg}}$, the off-diagonal block of the projected gradient, $-U_{\text{neg}}^\top \mathcal{G} V$, has entries $-u_{r+j}^\top \mathcal{G} v_i$—precisely the quantities weighted by $\mathcal{P}_{ij}$ and $\mathcal{Q}_{ij}$ in the curvature-coupling term of Theorem 3.1. These coefficients are generated implicitly by the gradient flow on the enlarged subspace, not tuned by hand. DLRA skips Step 2, so this block is identically zero and the coupling is lost.

Computational complexity. For the training of each layer, the first step—initializing K_0 and L_0—requires $O(nr^2)$ and $O(mr^2)$ operations, respectively, while the minimal computational cost of calculating the gradient flow with respect to K and L is $O(r(n+m))$. The second step incurs no computational cost. The third step (QR decomposition on the compensation matrix) requires $O(9nr^2)$ and $O(9mr^2)$ operations. The fourth step (initializing S_0) requires $O(3nr^2 + 3mr^2 + 6r^3)$ operations, with the minimal gradient update costing $O(3r(n+m) + 9r^2)$. The SVD computation in the worst case incurs a cost of $O(r^3)$, and computing the updated basis matrices for the new rank requires $O(4r^2(n+m))$. The tolerance adjustment in the sixth step incurs no computational cost. In summary, given that $r \ll \min(n, m)$, the computational cost for each layer can be controlled within $O(r^2(n+m))$, comparable to naive DLRT and significantly lower than the full-rank baseline's $O(nm)$.

3.4 Exactness Property and Error Analysis

In this section, we conduct a theoretical analysis of the proposed method. First, to demonstrate the exactness of our low-rank training, we assume the existence of a well-performing network with low-rank parameters under a certain initialization, and prove that our method can accurately learn these low-rank parameters.

Theorem 3.2 (Exactness)**.** *Assume that $W(t)$ represents the low-rank network weights obtained at time t through gradient-based training, starting from initialized weights $W(0)$. Under SDLRT, if the network is initialized as $W(0)$ and its parameters are factorized as $U(0), S(0), V(0)$, then after a certain training period of T, the following relationship holds: $U(t)S(t)V(t)^\top = W(t)$.*

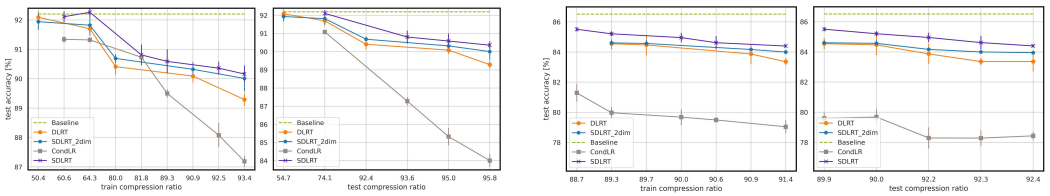
However, in the absence of additional constraints, networks trained via gradient-based methods may not exhibit significant low-rank properties. In practice, if the loss function is L-Lipschitz continuous and B-bounded, we can derive the following:

Theorem 3.3 (Error Bound)**.** *Assume that $W_f(t)$ represents the full-rank network weights obtained at time t through training based on Equation 1, starting from the initialized weights $W_f(0)$. Moreover, if $W_f(0)$ is projected onto a low-rank manifold as initialization, represented by $U(0), S(0), V(0)$, then after a training period of $T = n\eta$ where η denotes the time of each step, the SDLRT iterates $\hat{U}_t, \hat{S}_t, \hat{V}_t$ satisfy*

$$\|\hat{U}_t\hat{S}_t\hat{V}_t^\top - W_f(n\eta)\|_F \le c_0\delta + c_1\gamma\varepsilon + c_2\eta + c_3\vartheta/\eta,$$

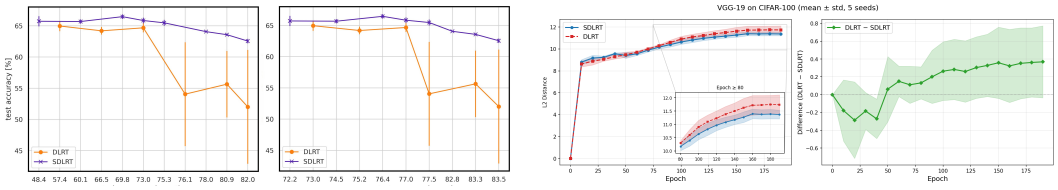
where c_0, c_1, c_2, c_3 depend only on L, B, T, $\delta = \|U_0S_0V_0^\top - W_f(0)\|_F$ is the initial projection error, and $\gamma \in (0, 1]$ is the spectral contraction factor induced by the compensation buffer. Subspace monotonicity always yields $\gamma \le 1$; moreover $\gamma < 1$ if and only if *the compensation basis captures nonzero gradient energy from the truncated directions (proved in Appendix C), which is exactly the aggressive-compression regime. In that regime the improvement over DLRT is strictly positive:*

$$E_{\text{DLRT}} - E_{\text{SDLRT}} = (1-\gamma)\, c_1\varepsilon \ > \ 0.$$



(a) VGG-16 on CIFAR-10 (b) AlexNet on CIFAR-10

Figure 3: Comparison of accuracy under different training and testing compression on CIFAR-10. SDLRT and $\text{SDLRT}_{\text{2dim}}$ achieve the top two highest scores. Error bars represent the minimum and maximum across 5 samples.



(a) VGG-19 on CIFAR-100 (b) L2 Distance of VGG-19 on CIFAR-100

Figure 4: On VGG-19/CIFAR-100, SDLRT can stably identify winning tickets under extreme compression rates, whereas DLRT cannot. Error bars represent the minimum and maximum across 5 samples.

4 Experiments

We evaluate SDLRT from three complementary perspectives. Section 4.1 examines its compression capability on image classification, showing that SDLRT stably identifies high-performing low-rank subnetworks even under extreme compression where existing methods collapse. Section 4.2 investigates training stability by tracking the weight-space distance between low-rank subnetworks and their full-rank counterparts throughout optimization, revealing that the proposed compensation mechanism yields consistently closer and less variable trajectories. Section 4.3 demonstrates the broad applicability of SDLRT by deploying it as a low-rank adapter for parameter-efficient fine-tuning of pretrained language models on SuperGLUE. The experiments are performed on an Nvidia RTX3080 and Nvidia RTX4090D 24GB. The code is available in the supplementary material.

Notation. A low-rank subnetwork is a tuple (W, r), where W represents the weights and r is the specified rank, as defined before. We write $\text{SVD}(W, r)$ for the best rank-r approximation of W. Suppose that, starting from the same initialization W_0: (i) full-rank training produces W_t at iteration t, and (ii) low-rank training initialized with $\text{SVD}(W_0, r_0)$ produces factors U_t, S_t, V_t. The distance at iteration t is $d_t = \|\text{SVD}(W_t, r_t) - U_t S_t V_t^\top\|_F$, and the stability of a low-rank training method is measured by d_T at convergence.

4.1 Image Classification under Aggressive Compression

Setup. We evaluate on CIFAR-10 (10 classes) and CIFAR-100 (100 classes), each with 50k training and 10k test images. We employ VGG-16 and AlexNet for CIFAR-10, and VGG-19 for CIFAR-100. We compare SDLRT against three methods: (i) the Baseline network trained with standard SGD at full rank; (ii) DLRT; (iii) CondLR [48], a low-rank training method that imposes approximate orthogonality constraints. To isolate the source of improvement, we additionally report $\text{SDLRT}_{\text{2dim}}$, a variant that applies the compensation directly to the original factors K and L using (K, U_{neg}) and (L, V_{neg}). Each configuration is evaluated over 5 independent runs with different random seeds. We train for 120 epochs with a mini-batch size of 128, initial learning rate 0.05 decayed by a factor of 0.1 at epochs 60 and 100, with momentum 0.1, and set the $\tau \in \{0.1, 0.2, 0.25, 0.3, 0.4\}$, $\omega = 0.8$.

Results. Since accuracy is largely preserved at low compression rates, we focus the analysis on the high-compression regime where performance differences become meaningful. **SDLRT discovers higher-reward winning tickets.** Figure 3 shows test accuracy on CIFAR-10 as the

Table 1: Component ablation of SDLRT on VGG-19/CIFAR-100 at 83% compression (test accuracy %, 3 seeds; min–max in brackets).

Buffer	Feedback	Test Acc. (%)	Range
–	–	56.60	$[54.53, 58.66]$
–	✓	59.71	$[58.44, 60.93]$
✓	–	64.04	$[63.49, 64.33]$
✓	✓	$\mathbf{64.90}$	$[64.75, 64.91]$

Table 2: Results (%) on fine-tuning DeBERTa-v3-base with SuperGLUE datasets.

Method	CB	COPA	WSC	RTE	WiC	BoolQ	Avg.
SDLRT (ours)	$83.63_{\pm 1.34}$	$91.00_{\pm 1.67}$	$\mathbf{91.51}_{\pm 1.13}$	$86.52_{\pm 0.59}$	$\mathbf{73.93}_{\pm 0.88}$	$84.29_{\pm 0.20}$	$\mathbf{85.15}$
DLRT	$81.85_{\pm 1.75}$	$\mathbf{91.17}_{\pm 0.98}$	$90.06_{\pm 1.32}$	$86.28_{\pm 1.07}$	$73.41_{\pm 0.61}$	$\mathbf{84.33}_{\pm 0.54}$	84.52
LoRA	$\mathbf{85.72}_{\pm 2.26}$	$90.50_{\pm 0.55}$	$84.93_{\pm 10.53}$	$\mathbf{87.30}_{\pm 0.93}$	$73.07_{\pm 0.81}$	$84.14_{\pm 0.21}$	84.28
AdaLoRA	$82.74_{\pm 2.16}$	$90.67_{\pm 2.07}$	$63.46_{\pm 0.00}$	$85.92_{\pm 0.82}$	$73.49_{\pm 0.54}$	$83.92_{\pm 0.14}$	80.03
LoRA+	$85.12_{\pm 2.69}$	$89.83_{\pm 1.83}$	$89.58_{\pm 2.23}$	$87.24_{\pm 1.16}$	$73.27_{\pm 0.57}$	$83.80_{\pm 0.56}$	84.81

compression rate increases. At moderate compression all methods perform comparably, and SDLRT's compensation overhead can even introduce mild overfitting. However, once the compression rate exceeds $\approx 80\%$, a clear separation emerges: SDLRT maintains accuracy above 90% on VGG-16 and above 84% on AlexNet, while DLRT and CondLR degrade substantially. The comparison between $\mathrm{SDLRT}_{2\mathrm{dim}}$ and the DLRT group under the same dimensional settings directly demonstrates the superiority of the proposed algorithm. **SDLRT stably identifies winning tickets under extreme compression.** Figure 4a compares VGG-19 on CIFAR-100 at progressively higher compression rates. When compression exceeds 76.1%, DLRT may fail to identify trainable low-rank subnetworks: accuracy collapses and variance across seeds explodes. SDLRT, in contrast, continues to discover stable winning tickets by supplementing discarded singular value information during factor updates, demonstrating the effectiveness of the compensation mechanism. **Component ablation.** Table 1 disentangles the two mechanisms on VGG-19/CIFAR-100 at 83% compression. Relative to DLRT (56.60%), the buffer and the feedback contribute $+7.4$ and $+3.1$ points in isolation and are nearly additive in combination (64.90%, $+8.3$); the full model also attains the smallest seed-to-seed spread (0.16 vs. 0.84). The buffer drives most of the accuracy gain, while the feedback chiefly stabilizes training. **Comparison with spectral regularization.** SDLRT also outperforms RobustDLRT [49], a spectral-regularization baseline, by a wide margin (64.20% vs. 57.15% at 83% compression): regularizing the conditioning of S does not prevent rank collapse, leaving RobustDLRT statistically indistinguishable from DLRT. Full results are given in Appendix D.4.

4.2 Training Stability of Low-Rank Subnetworks

Beyond final accuracy, a practically important question is *how closely does a low-rank training trajectory track the full-rank solution throughout optimization?* A smaller and more consistent distance d_t implies that the low-rank method faithfully approximates the expressive capacity of the full network, reducing the risk of divergent training dynamics.

Setup. Using the same model architectures and datasets as Section 4.1 (VGG-19 on CIFAR-100), we set $\tau = 0.4$ and record the distance d_t between the low-rank subnetwork and the full-rank baseline at every epoch for both DLRT and SDLRT. We report the mean and standard deviation across 5 seeds.

Results. Figure 4b plots d_t over the course of training. **SDLRT converges to a closer neighborhood of the full-rank solution.** At convergence, the distance d_T of SDLRT is markedly smaller than that of DLRT. On VGG-19 at $\tau = 0.4$, Figure 4b shows that the terminal distance is reduced by approximately 10% relative to DLRT. We additionally measure the contraction factor γ of Theorem 3.3 directly (Appendix D.3): it lies in $[0.71, 0.99]$ with mean 0.93 across layers and epochs, empirically confirming $\gamma < 1$ under aggressive compression.

4.3 Parameter-Efficient Fine-Tuning on SuperGLUE

To demonstrate that SDLRT extends naturally beyond from-scratch training, we apply it as a low-rank adapter for parameter-efficient fine-tuning (PEFT) of pretrained language models.

Setup. We fine-tune DeBERTa-v3-base [50] on six SuperGLUE [51] tasks: BoolQ, CB, COPA, RTE, WiC, and WSC. These tasks span a broad spectrum of NLU capabilities, from surface-level matching to nuanced commonsense reasoning, providing a comprehensive testbed for adapter quality. We compare with three widely adopted low-rank PEFT methods: LoRA [52], which injects fixed-rank decomposition matrices into frozen pretrained weights; LoRA+ [53], which assigns different learning rates to the two low-rank factors; and AdaLoRA [43], which adaptively allocates rank budgets across layers via importance scoring. For all methods, low-rank adaptation is applied to the query, key and value projection matrices of every self-attention layer. LoRA, LoRA+ and AdaLoRA use an initial rank $r = 10$, resulting in ≈ 1.145M trainable parameters. DLRT and SDLRT start at $r = 10$ with singular-value tolerance $\tau = 0.02$, yielding ≈ 1.159M and ≈ 1.177M ($\approx 2.8\%$ overhead as the upper bound) trainable parameters respectively. All experiments are repeated over 6 random seeds. Six tasks are conducted for $\{30, 20, 20, 20, 30, 30\}$ epochs with batch size 16, learning rate 6×10^{-4} with linear warm-up over the first 6% of steps and weight decay 0.01; for LoRA+, we set the learning rate to 2×10^{-4} and the learning-rate ratio to 8.

Results. Table 2 summarizes validation accuracy across six SuperGLUE tasks. As shown in Table 2, SDLRT achieves the highest average accuracy of 85.15%, with gains of at least 0.34%. We further confirm that this advantage is not a parameter-budget artifact (Appendix D.6) and that it carries over to a billion-parameter model, Qwen2.5-3B (Appendix D.7).

We note that AdaLoRA collapses on WSC to 63.46 ± 0.00. The zero variance across seeds corresponds to every run degenerating to the majority-class predictor (63.46% is the WSC majority-class rate): this is a genuine failure mode of importance-based rank reallocation on this small, class-imbalanced task, not a reporting artifact.

5 Discussion and Limitations

This work first derives the gradient-flow equations for the SVD factorization of full-parameter and low-rank network weights based on the manifold's geometric properties, and identifies that the difference between the two is related to the interaction between the small singular-value components and the large ones. Based on this, we propose a training method that enhances the stability of low-rank networks. The proposed method does not increase the rank of the network under a given tolerance but incorporates information from small singular values, thereby maintaining the accuracy of low-rank networks even under extreme compression rates. Moreover, the compensatory low-rank training framework proposed in this paper may provide insights for other low-rank training methods under different regularization settings. However, we note that the negative-feedback factor ω is set heuristically and scaling SDLRT to large architectures remains future work.

Acknowledgments and Disclosure of Funding

The authors received no specific funding for this work. We thank the anonymous reviewers for their constructive feedback.

References

[1] Albert Gural, Phillip Nadeau, Mehul Tikekar, and Boris Murmann. Low-rank training of deep neural networks for emerging memory technology, 2021. URL `https://arxiv.org/abs/2009.03887`.

[2] Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. In *NeurIPS*, 2022.

[3] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*. OpenReview.net, 2019.

[4] Haoran You, Baopu Li, Zhanyi Sun, Xu Ouyang, and Yingyan Lin. Supertickets: Drawing task-agnostic lottery tickets from supernets via jointly architecture searching and parameter pruning. In *ECCV (11)*, Lecture Notes in Computer Science, pages 674–690. Springer, 2022.

[5] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *ICML*, Proceedings of Machine Learning Research, pages 3259–3269. PMLR, 2020.

[6] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis, 2020. URL `https://arxiv.org/abs/1903.01611`.

[7] Othmar Koch and Christian Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.

[8] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016. URL `https://arxiv.org/abs/1510.00149`.

[9] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *ICML*, Proceedings of Machine Learning Research, pages 10323–10337. PMLR, 2023.

[10] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *ICLR*. OpenReview.net, 2024.

[11] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *CVPR*, pages 16091–16101. IEEE, 2023.

[12] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *NeurIPS*, 2023.

[13] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *ICLR*. OpenReview.net, 2024.

[14] Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(5):2900–2919, 2024.

[15] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR (Poster)*. OpenReview.net, 2019.

[16] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *ICLR*. OpenReview.net, 2020.

[17] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):10558–10578, 2024.

[18] Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *ICML*, Proceedings of Machine Learning Research, pages 20336–20350. PMLR, 2023.

[19] Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *ACL (Findings)*, Findings of ACL, pages 3013–3026. Association for Computational Linguistics, 2024.

[20] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán. Low-rank compression of neural nets: Learning the rank of each layer. In *CVPR*, pages 8046–8056. Computer Vision Foundation / IEEE, 2020.

[21] Ari S. Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *NeurIPS*, pages 4933–4943, 2019.

[22] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Michael Carbin, and Zhangyang Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In *CVPR*, pages 16306–16316. Computer Vision Foundation / IEEE, 2021.

[23] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained BERT networks. In *NeurIPS*, 2020.

[24] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *CVPR*, pages 11890–11899. Computer Vision Foundation / IEEE, 2020.

[25] Eran Malach, Gilad Yehudai, Shai Shalev-Shwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *ICML*, Proceedings of Machine Learning Research, pages 6682–6691. PMLR, 2020.

[26] Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris S. Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. In *NeurIPS*, 2020.

[27] Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the lottery ticket hypothesis for convolutional neural networks. In *ICLR*. OpenReview.net, 2022.

[28] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *ICML*, Proceedings of Machine Learning Research, pages 1695–1706. PMLR, 2021.

[29] Bo Hui, Da Yan, Xiaolong Ma, and Wei-Shinn Ku. Rethinking graph lottery tickets: Graph sparsity matters. In *ICLR*. OpenReview.net, 2023.

[30] Xuxi Chen, Zhenyu Zhang, Yongduo Sui, and Tianlong Chen. Gans can play lottery tickets too. In *ICLR*. OpenReview.net, 2021.

[31] Chao Jiang, Bo Hui, Bohan Liu, and Da Yan. Successfully applying lottery ticket hypothesis to diffusion model, 2023. URL `https://arxiv.org/abs/2310.18823`.

[32] Bohan Liu, Zijie Zhang, Peixiong He, Zhensen Wang, Yang Xiao, Ruimeng Ye, Yang Zhou, Wei-Shinn Ku, and Bo Hui. A survey of lottery ticket hypothesis, 2024. URL `https://arxiv.org/abs/2403.04861`.

[33] Mansheej Paul, Feng Chen, Brett W. Larsen, Jonathan Frankle, Surya Ganguli, and Gintare Karolina Dziugaite. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? In *ICLR*. OpenReview.net, 2023.

[34] Zeru Zhang, Jiayin Jin, Zijie Zhang, Yang Zhou, Xin Zhao, Jiaxiang Ren, Ji Liu, Lingfei Wu, Ruoming Jin, and Dejing Dou. Validating the lottery ticket hypothesis with inertial manifold theory. In *NeurIPS*, pages 30196–30210, 2021.

[35] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, and Yanzhi Wang. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? In *NeurIPS*, pages 12749–12760, 2021.

[36] Rebekka Burkholz. Convolutional and residual networks provably contain lottery tickets. In *ICML*, Proceedings of Machine Learning Research, pages 2414–2433. PMLR, 2022.

[37] Gianluca Ceruti and Christian Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62:23–44, 2022.

[38] Gianluca Ceruti, Jonas Kusch, and Christian Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62:1149–1174, 2022.

[39] Emanuele Zangrando, Steffen Schotthöfer, Gianluca Ceruti, Jonas Kusch, and Francesco Tudisco. Geometry-aware training of factorized layers in tensor tucker format. In *NeurIPS*, 2024.

[40] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *NeurIPS*, pages 7411–7422, 2019.

[41] Tomer Galanti, Zachary S. Siegel, Aparna Gupte, and Tomaso Poggio. SGD and weight decay secretly minimize the rank of your neural network, 2024. URL `https://arxiv.org/abs/2206.05794`.

[42] Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *CVPR Workshops*, pages 2899–2908. Computer Vision Foundation / IEEE, 2020.

[43] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*. OpenReview.net, 2023.

[44] Mikhail Khodak, Neil A. Tenenholtz, Lester Mackey, and Nicolò Fusi. Initialization and regularization of factorized neural layers. In *ICLR*. OpenReview.net, 2021.

[45] Noam Razin, Asaf Maman, and Nadav Cohen. Implicit regularization in hierarchical tensor factorization and deep convolutional neural networks. In *ICML*, Proceedings of Machine Learning Research, pages 18422–18462. PMLR, 2022.

[46] Florian Feppon and Pierre F. J. Lermusiaux. A geometric approach to dynamical model order reduction. *SIAM J. Matrix Anal. Appl.*, 39(1):510–538, 2018.

[47] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *ICML*, Proceedings of Machine Learning Research, pages 61121–61143. PMLR / OpenReview.net, 2024.

[48] Dayana Savostianova, Emanuele Zangrando, Gianluca Ceruti, and Francesco Tudisco. Robust low-rank training via approximate orthonormal constraints. In *NeurIPS*, 2023.

[49] Steffen Schotthöfer, Lexie Yang, and Stefan Schnake. Dynamical low-rank compression of neural networks with robustness under adversarial attacks. In *NeurIPS*, 2025.

[50] Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using electra-style pre-training with gradient-disentangled embedding sharing. In *ICLR*. OpenReview.net, 2023.

[51] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, pages 3261–3275, 2019.

[52] 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. In *ICLR*. OpenReview.net, 2022.

[53] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *ICML*, Proceedings of Machine Learning Research, pages 17783–17806. PMLR / OpenReview.net, 2024.

[54] Ernst Hairer, Syvert Paul Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2 edition, 1993.

A Broader Impact Statement

This paper presents work whose main goal is to improve the stability and reliability of low-rank neural network training under aggressive compression, while maintaining mathematically sound guarantees of convergence and approximation. As in the majority of deep learning research, there are potential societal consequences of our work, none of which we feel must be specifically highlighted here. A positive societal impact is the reduced carbon emissions and energy consumption enabled by more efficient neural network training and inference through effective model compression. Regarding ethical aspects, we feel nothing has to be added.

B Proof of Theorem 3.2 (Exactness)

We show that, when the right-hand side $F(t, Y) = \dot{W}(t)$ corresponds to a solution $W(t)$ of exact rank r on $[t_0, t_1]$, the SDLRT integrator reproduces $W(t_1)$ exactly. The argument adapts the exactness proof of the rank-adaptive integrator in [38] to account for the additional compensation bases U_{neg} and V_{neg}.

Proof. Let $W(t) = U(t)S(t)V(t)^\top$ with $\mathrm{rank}(W(t)) = r$ for all $t \in [t_0, t_1]$, and set $Y_0 = U_0 S_0 V_0^\top = W(t_0)$.

K-step and L-step. The K-step ODE $\dot{K} = \dot{W}(t)\, V_0$ with $K(t_0) = U_0 S_0$ integrates to

$$K_1 = U_0 S_0 + (W(t_1) - W(t_0))V_0 = W(t_1)\, V_0. \tag{19}$$

Analogously, $L_1 = W(t_1)^\top U_0$.

Compensated basis augmentation. SDLRT forms the augmented matrices $(K_1, U_0, U_{\mathrm{neg}})$ and $(L_1, V_0, V_{\mathrm{neg}})$, and computes their QR factorizations to obtain orthonormal bases $\widehat{U}$ and $\widehat{V}$.

Since $W(t_1) = U(t_1)S(t_1)V(t_1)^\top$ with $S(t_1)$ non-singular and $V(t_1)^\top V_0$ invertible (by assumption), Equation (19) gives $\mathrm{range}(K_1) = \mathrm{range}(U(t_1)) = \mathrm{range}(W(t_1))$. Because $\mathrm{range}(W(t_1)) \subseteq \mathrm{range}(K_1, U_0, U_{\mathrm{neg}})$, we have

$$\widehat{U}\widehat{U}^\top W(t_1) = W(t_1). \tag{20}$$

An identical argument yields $W(t_1)\widehat{V}\widehat{V}^\top = W(t_1)$. Note that the compensation columns U_{neg} (resp. V_{neg}) only enlarge the column space of the augmented matrix; they therefore preserve inclusion (20) and cannot compromise exactness.

S-step (Galerkin update). Because U_0 is a sub-block of $(K_1, U_0, U_{\mathrm{neg}})$, we have $\widehat{U}\widehat{U}^\top U_0 = U_0$ (likewise $\widehat{V}\widehat{V}^\top V_0 = V_0$). Setting $\widehat{M} = \widehat{U}^\top U_0$ and $\widehat{N} = \widehat{V}^\top V_0$, the S-step initial value is

$$\widehat{S}(t_0) = \widehat{M}\, S_0\, \widehat{N}^\top = \widehat{U}^\top\, W(t_0)\, \widehat{V}, \tag{21}$$

so the augmented initial approximation satisfies $\widehat{U}\widehat{S}(t_0)\widehat{V}^\top = W(t_0) = Y_0$, confirming that the compensation does not alter the starting point.

Integrating the S-step ODE $\dot{\widehat{S}} = \widehat{U}^\top \dot{W}(t)\widehat{V}$ gives

$$\widehat{S}(t_1) = \widehat{U}^\top W(t_0)\widehat{V} + \widehat{U}^\top (W(t_1) - W(t_0))\widehat{V} = \widehat{U}^\top W(t_1)\widehat{V}. \tag{22}$$

Combining with (20) and its right counterpart:

$$\widehat{U}\, \widehat{S}(t_1)\, \widehat{V}^\top = \widehat{U}\widehat{U}^\top W(t_1)\widehat{V}\widehat{V}^\top = W(t_1). \tag{23}$$

Truncation. Since $\mathrm{rank}(W(t_1)) = r$, the matrix $\widehat{S}(t_1)$ has exactly r non-zero singular values. Choosing the tolerance ϑ ensures that truncation retains all r components, yielding $Y_1 = U_1 S_1 V_1^\top = W(t_1)$. □

C Proof of Theorem 3.3 (Error Bound)

Setup and Assumptions. Let $W_f(t)$ denote the full-parameter weight evolved by the gradient flow

$$\dot{W}_f(t) = \mathcal{F}(W_f(t)) := -\nabla_W \mathcal{L}(W_f(t)), \qquad W_f(0) \text{ given.}$$

We make the following assumptions, identical to those of DLRT [2]:

(A1) *Boundedness and Lipschitz continuity.* There exist constants $B, L > 0$ such that, for every $W, \widetilde{W}$, $\|\mathcal{F}(W)\|_F \le B$ and $\|\mathcal{F}(W) - \mathcal{F}(\widetilde{W})\|_F \le L\|W - \widetilde{W}\|_F$.

(A2) *ε-closeness to the manifold.* For every $W \in \mathcal{M}_r$ in a neighborhood of $W_f(t)$ along the flow, $\|(I - \Pi_{\mathcal{T}(W)})\,\mathcal{F}(W)\|_F \le \varepsilon$.

(A3) *Initial projection error.* The SDLRT initialization satisfies $\|U(0)S(0)V(0)^\top - W_f(0)\|_F \le \delta$.

Theorem C.1 (Error Bound)**.** *Let $\widehat{U}_t, \widehat{S}_t, \widehat{V}_t$ denote the SDLRT iterates after n steps of size $\eta > 0$, with singular-value truncation tolerance $\vartheta > 0$. Under assumptions (A1)–(A3), there exist constants $c_0, c_1, c_2, c_3 > 0$ depending only on L, B and the final time $T = n\eta$ (in particular, independent of any singular value of the exact or approximate solutions) such that*

$$\big\|\widehat{U}_t\widehat{S}_t\widehat{V}_t^\top - W_f(n\eta)\big\|_F \;\le\; c_0\,\delta + c_1\,\gamma\varepsilon + c_2\,\eta + c_3\,\vartheta/\eta.$$

Proof. We adapt the four-lemma argument of DLRT to the SDLRT compensation step. Throughout the proof, $A(t)$ denotes the exact local solution of $\dot{A} = \mathcal{F}(A)$ on $[0, \eta]$ with starting value $Y_0 = U_0 S_0 V_0^\top$, and $A_1 := A(\eta)$.

Step 1: Compensation gain. The augmented bases updated by DLRT and SDLRT respectively are:

$$\begin{cases} \widetilde{U} \in \text{orth}(K_1, U), \quad \widetilde{V} \in \text{orth}(L_1, V) \\ \widehat{U} \in \text{orth}(K_1, U, U_{\text{neg}}), \quad \widehat{V} \in \text{orth}(L_1, V, V_{\text{neg}}). \end{cases} \tag{24}$$

Thus, there is a strict subspace inclusion relationship:

$$\text{range}(\widetilde{U}) \subset \text{range}(\widehat{U}), \quad \text{range}(\widetilde{V}) \subset \text{range}(\widehat{V}). \tag{25}$$

Therefore, by the monotonicity of the orthogonal projection onto a larger subspace (projecting onto $\text{range}(\widehat{U}) \supseteq \text{range}(\widetilde{U})$ cannot increase the residual), there exists $\gamma \in (0, 1]$ such that

$$\begin{cases} \|(I - \widehat{U}\widehat{U}^\top)A_1\|_F \le \gamma\|(I - \widetilde{U}\widetilde{U}^\top)A_1\|_F \le \gamma\theta \\ \|(I - \widehat{V}\widehat{V}^\top)A_1\|_F \le \gamma\|(I - \widetilde{V}\widetilde{V}^\top)A_1\|_F \le \gamma\theta \end{cases} \tag{26}$$

where A_1 is the time-η value of the local exact flow with starting value Y_0, and θ is the DLRT projection bound

$$\theta \;=\; BL\big(4e^{L\eta} + 9\big)\eta^2 + \big(3e^{L\eta} + 4\big)\varepsilon\eta + e^{L\eta}\delta_{\text{loc}},$$

with δ_{loc} the deviation between the local starting iterate and the full-rank flow.

Non-increase and strictness of γ. Let $P_{\widehat{U}} = \widehat{U}\widehat{U}^\top$ and $P_{\widetilde{U}} = \widetilde{U}\widetilde{U}^\top$ be the orthogonal projectors onto the SDLRT and DLRT augmented left bases. Since $\text{range}(\widetilde{U}) = \text{range}[K_1, U] \subseteq \text{range}[K_1, U, U_{\text{neg}}] = \text{range}(\widehat{U})$, the difference $P_{\widehat{U}} - P_{\widetilde{U}}$ is itself an orthogonal projector, hence for any matrix M

$$\|(I - P_{\widehat{U}})M\|_F^2 = \|(I - P_{\widetilde{U}})M\|_F^2 - \|(P_{\widehat{U}} - P_{\widetilde{U}})M\|_F^2 \le \|(I - P_{\widetilde{U}})M\|_F^2. \tag{27}$$

Taking $M = A_1$ (and its transpose for the right basis) yields inequalities (26) with $\gamma \le 1$. Equality $\gamma = 1$ holds *only* when $(P_{\widehat{U}} - P_{\widetilde{U}})A_1 = 0$, i.e. when U_{neg} is orthogonal to the truncated residual $(I - P_{\widetilde{U}})A_1$. Because U_{neg} stores exactly the top discarded singular directions of the previous iterate, this inner product is nonzero whenever those directions carry nonzero gradient energy—precisely the aggressive-compression regime where the spectral gap $\sigma_i^2 - \sigma_{r+j}^2$ is small (Theorem 3.1). Hence $\gamma < 1$ strictly in that regime.

Remark (staleness of the buffer). The compensation basis $U_{\mathrm{neg}}, V_{\mathrm{neg}}$ is taken from the *previous* step, because the directions truncated at the current step do not exist until after the augmentation—this is a causal necessity, not an approximation choice. By the Wedin $\sin\Theta$ theorem the induced basis perturbation is $O(\eta)$ over one step, so using the one-step-stale estimate perturbs the bound only at order $O(\eta)$; it is absorbed into the $c_2\eta$ term and does not affect the $c_1\gamma\varepsilon$ improvement.

Step 2: Two-sided compensated projection error. From (26) and the unit-norm property of orthogonal projectors,

$$\begin{aligned}\big\|\widehat{U}\widehat{U}^\top A_1\widehat{V}\widehat{V}^\top - A_1\big\|_F &\le \big\|(\widehat{U}\widehat{U}^\top A_1 - A_1)\widehat{V}\widehat{V}^\top\big\|_F + \big\|A_1\widehat{V}\widehat{V}^\top - A_1\big\|_F \\ &\le \|(I-\widehat{U}\widehat{U}^\top)A_1\|_F\cdot\|\widehat{V}\widehat{V}^\top\|_2 + \|(I-\widehat{V}\widehat{V}^\top)A_1^\top\|_F \\ &\le 2\gamma\theta.\end{aligned} \tag{28}$$

This is the SDLRT analogue of DLRT's Lemma 3 (Lemma 6.2 in the appendix), with the bound reduced by the factor $\gamma \le 1$ (strictly < 1 under the condition established above).

Step 3: Auxiliary decomposition for the S-step. Define $\widetilde{S}(t) := \widehat{U}^\top A(t)\widehat{V}$ for $t \in [0,\eta]$, and decompose

$$A(t) \;=\; \widehat{U}\widetilde{S}(t)\widehat{V}^\top + \mathcal{R}(t), \qquad \mathcal{R}(t) := A(t) - \widehat{U}\widehat{U}^\top A(t)\widehat{V}\widehat{V}^\top.$$

By (A1) and (28),

$$\|\mathcal{R}(t)\|_F \le \|\mathcal{R}(t)-\mathcal{R}(\eta)\|_F + \|\mathcal{R}(\eta)\|_F \le 2B\eta + 2\gamma\theta. \tag{29}$$

Writing $\mathcal{F}(A(t)) = \mathcal{F}(\widehat{U}\widetilde{S}(t)\widehat{V}^\top) + \mathcal{D}(t)$ with defect $\mathcal{D}(t)$, the Lipschitz bound (A1) yields

$$\|\mathcal{D}(t)\|_F \le L\,\|\mathcal{R}(t)\|_F \le 2L\big(B\eta + \gamma\theta\big). \tag{30}$$

Step 4: Galerkin S-step error via Gronwall. The SDLRT S-step ODE

$$\dot{\widehat{S}}(t) = \widehat{U}^\top\mathcal{F}(\widehat{U}\widehat{S}(t)\widehat{V}^\top)\widehat{V}, \qquad \widehat{S}(0) = \widehat{U}^\top Y_0\widehat{V},$$

and the auxiliary equation

$$\dot{\widetilde{S}}(t) = \widehat{U}^\top\mathcal{F}(\widehat{U}\widetilde{S}(t)\widehat{V}^\top)\widehat{V} + \widehat{U}^\top\mathcal{D}(t)\widehat{V}, \qquad \widetilde{S}(0) = \widehat{U}^\top Y_0\widehat{V},$$

share the same initial value (since $A_0 = Y_0$ locally). The Gronwall inequality combined with (30) gives

$$\big\|\widehat{S}(\eta) - \widehat{U}^\top A_1\widehat{V}\big\|_F \le \int_0^\eta e^{L(\eta-s)}\|\mathcal{D}(s)\|_F\,ds \le 2Le^{L\eta}\big(B\eta+\gamma\theta\big)\eta. \tag{31}$$

Step 5: Local error of one SDLRT step. Setting $Y_1 := \widehat{U}\widehat{S}(\eta)\widehat{V}^\top$ and combining (28) with (31),

$$\begin{aligned}\|Y_1 - A_1\|_F &\le \big\|\widehat{U}\widehat{S}(\eta)\widehat{V}^\top - \widehat{U}\widehat{U}^\top A_1\widehat{V}\widehat{V}^\top\big\|_F + \big\|\widehat{U}\widehat{U}^\top A_1\widehat{V}\widehat{V}^\top - A_1\big\|_F \\ &\le \big\|\widehat{S}(\eta) - \widehat{U}^\top A_1\widehat{V}\big\|_F + 2\gamma\theta \\ &\le 2Le^{L\eta}B\eta^2 + 2Le^{L\eta}\gamma\theta\eta + 2\gamma\theta.\end{aligned} \tag{32}$$

Using the explicit form of θ (with $\delta_{\mathrm{loc}} = 0$ for the per-step analysis),

$$\gamma\theta = \gamma\big[BL(4e^{L\eta}+9)\eta^2 + (3e^{L\eta}+4)\varepsilon\eta\big],$$

and absorbing all L, B-dependent factors into constants $\widehat{c}_1, \widehat{c}_2$ that depend only on L, B, η, the bound (32) simplifies to

$$\|Y_1 - A_1\|_F \;\le\; \widehat{c}_1\,\gamma\varepsilon\,\eta + \widehat{c}_2\,\eta^2. \tag{33}$$

Step 6: Truncation step. After the rank-adaptive truncation in SDLRT, the discarded tail contributes at most ϑ in Frobenius norm. Writing $Y_1^{\mathrm{trunc}} = \widehat{U}_1\widehat{S}_1\widehat{V}_1^\top$ and applying the triangle inequality to (33),

$$\|Y_1^{\mathrm{trunc}} - A_1\|_F \;\le\; \widehat{c}_1\,\gamma\varepsilon\,\eta + \widehat{c}_2\,\eta^2 + \vartheta. \tag{34}$$

Step 7: Global error via Lady Windermere's fan. By Assumption (A1), the flow map of $\dot{W} = \mathcal{F}(W)$ is $e^{L\eta}$-Lipschitz over each step. Applying the standard error-propagation argument [54, Sec. II.3] to the per-step error (34) over n steps with $T = n\eta$,

$$\begin{aligned}\big\|\widehat{U}_t\widehat{S}_t\widehat{V}_t^\top - W_f(n\eta)\big\|_F &\le e^{LT}\,\delta + \sum_{k=0}^{n-1} e^{L(n-k-1)\eta}\big(\widehat{c}_1\gamma\varepsilon\,\eta + \widehat{c}_2\,\eta^2 + \vartheta\big) \\ &\le e^{LT}\,\delta + n\,e^{LT}\big(\widehat{c}_1\gamma\varepsilon\,\eta + \widehat{c}_2\,\eta^2 + \vartheta\big) \\ &= e^{LT}\,\delta + e^{LT}\widehat{c}_1\,T\,\gamma\varepsilon + e^{LT}\widehat{c}_2\,T\,\eta + e^{LT}\,T\,\vartheta/\eta.\end{aligned} \tag{35}$$

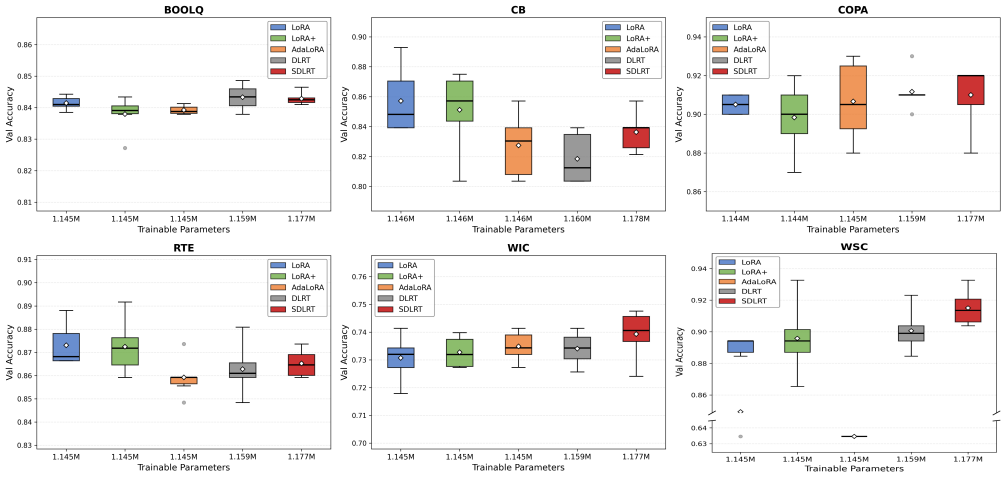


Figure 5: Distribution of the validation accuracy on DeBERTa-v3-base across six SuperGLUE tasks over six random seeds. The diamond marker indicates the mean and the circular marker indicates outliers. The box shows the interquartile range (25%–75% quantiles) and the median.

Table 3: Comparison of memory and time usage.

Method	Memory	Training Time (Average per epoch)	Test Accuracy(%)
DLRT	2680MB	56.88s	53.87%
SDLRT	2710MB	57.53s	61.86%

Setting

$$c_0 = e^{LT}, \qquad c_1 = e^{LT}\widehat{c}_1\,T, \qquad c_2 = e^{LT}\widehat{c}_2\,T, \qquad c_3 = e^{LT}\,T,$$

we obtain

$$\left\|\widehat{U}_t\widehat{S}_t\widehat{V}_t^{\top} - W_f(n\eta)\right\|_F \;\le\; c_0\,\delta + c_1\,\gamma\varepsilon + c_2\,\eta + c_3\,\vartheta/\eta,$$

where c_0, c_1, c_2, c_3 depend only on L, B, T and are independent of the singular values of any iterate. □ □

D Additional experiments

D.1 Additional experiments for the distribution of Table 2

Figure 5 shows the distribution of the validation accuracy from Table 2 as a supplement to the results in section 4.3.

D.2 Additional experiments for the memory and time cost

In Table 3, we report the memory and time usage on VGG-19/CIFAR-100 setting seed $= 30$, $\tau = 0.45$ and batch size 128. As in Section 4.1, the training is running on an NVIDIA RTX3080.

D.3 Empirical measurement of the contraction factor γ

To validate the mechanism behind Theorem 3.3, we estimate $\gamma = \|(I - P_{\widehat{U}})A_1\|_F / \|(I - P_{\widetilde{U}})A_1\|_F$ per layer, using the reconstructed weight at the *next* checkpoint as a proxy for the local exact flow A_1 (VGG-19/CIFAR-100, 83% compression). Across all layers and epochs $\gamma \in [0.71, 0.99]$ with mean 0.93; the strongest contraction ($\gamma \approx 0.71$) occurs in early training, when the truncated tail carries the most energy. Every measured value satisfies $\gamma < 1$, consistent with the strict-improvement condition of Appendix C.

D.4 Comparison with RobustDLRT (spectral regularization)

We compare against RobustDLRT [49], which augments the S-step with the spectral regularizer $R(S) = \|S^\top S - \alpha^2 I\|_F$ ($\beta = 10^{-3}$, 5 seeds, VGG-19/CIFAR-100 at 83% compression).

Table 4: Comparison with RobustDLRT at 83% compression (test accuracy %, 5 seeds; min–max in brackets).

Method	Test Acc. (%)	Range
DLRT	56.68	[54.57, 58.84]
RobustDLRT	57.15	[51.01, 61.63]
SDLRT (ours)	**64.20**	[63.78, 64.63]
SDLRT + spectral reg.	61.04	[59.78, 62.98]

RobustDLRT matches DLRT in mean accuracy with even larger variance: spectral regularization controls the conditioning of S but does not stop the rank from collapsing under aggressive compression, and yields no clean-accuracy benefit here. The stacked SDLRT + spectral variant is slightly lower, as expected—the regularizer trades clean accuracy for adversarial robustness—confirming that the two mechanisms are orthogonal and can be composed.

D.5 Sensitivity to ω, buffer size, and tolerance τ

We sweep the three hyperparameters on VGG-19/CIFAR-100 (2 seeds). These runs use a $\approx 71\%$ working point, so the absolute numbers are not directly comparable to Tables 1–4; only the trends matter. **(i) Feedback factor ω:** accuracy is on a plateau for $\omega \in [0.6, 0.9]$ (65.4–66.7) and drops to 60.2 at $\omega = 1.0$ (feedback off) with variance exploding, confirming feedback as a stabilizer. **(ii) Buffer size:** $0.5r \to 66.2$, $1.0r \to 66.7$—the gain saturates by $1.0r$. **(iii) Tolerance τ:** monotonically controls the converged rank (τ=0.2 $\to$ rank 36; τ=0.45 $\to$ rank 7.2), with best accuracy at $\tau = 0.4$.

D.6 Matched-budget comparison

To verify that the accuracy advantage of SDLRT is not merely an artifact of a larger parameter budget, we control the budget in *both* directions on DeBERTa-v3-base (six SuperGLUE tasks, 6 seeds, best-validation epoch). We report two parameter counts: the *trainable* parameters during fine-tuning, and the *deployed adapter*, i.e. the trained low-rank factors $USV^\top$ (BA for LoRA) retained at the end of training, excluding the classification head. "DLRT-rand ($3r$)" augments DLRT to the same $3r$ subspace dimension as SDLRT but fills the extra directions with random orthonormal vectors instead of the compensation buffer.

Table 5: Matched-budget comparison on SuperGLUE (DeBERTa-v3-base, 6 seeds). The deployed adapter is reported in absolute terms and as a multiple of LoRA's.

Method	Budget control	Trainable params	Deployed adapter	Avg. Acc. (%)
SDLRT (r_0=10)	default (reference)	1,177,490	0.363M (0.66×)	**85.15**
SDLRT (r_0=9)	shrunk below LoRA's budget	1,116,038	0.337M (0.61×)	85.09
DLRT (r_0=10)	default	1,159,490	0.363M (0.66×)	84.52
DLRT-rand ($3r$)	raised to SDLRT's budget	1,177,490	0.363M (0.66×)	84.69
LoRA (r=10)	default	1,145,090	0.553M (1.00×)	84.28
LoRA (r=11)	raised above SDLRT's budget	1,200,386	0.608M (1.10×)	84.31

The advantage of SDLRT persists under both controls. Shrinking SDLRT below LoRA's trainable budget (r_0=9) still yields 85.09%, above every baseline; conversely, raising DLRT and LoRA to (or above) SDLRT's budget leaves them at 84.69% and 84.31%, well below SDLRT. Notably, DLRT-rand consumes exactly the same $3r$ budget as SDLRT yet gains little, isolating the effect to *what* fills the enlarged subspace: random directions carry no curvature energy from the truncated tail, whereas the compensation buffer does—exactly the condition under which the contraction factor satisfies $\gamma < 1$ (Theorem 3.3). Finally, despite a comparable training budget, SDLRT's deployed adapter is only $0.66\times$ the size of LoRA's, as rank truncation produces a lower effective rank at convergence.

D.7 Scaling to a billion-parameter model

To test whether SDLRT scales beyond base-sized encoders, we fine-tune Qwen2.5-3B, inserting adapters on the query/key/value projections of all 36 layers (bf16 base, fp32 adapters). We train for 20 epochs with batch size 16, learning rate 3×10^{-4}, tolerance $\tau = 0.02$ and feedback interval 20 (matching the DeBERTa setup), over seeds $\{30, 100, 123\}$.

Table 6: Fine-tuning Qwen2.5-3B on three SuperGLUE tasks (accuracy %, mean±std over 3 seeds). The deployed adapter is reported as a multiple of LoRA's.

Method	RTE	WiC	BoolQ	Deployed adapter
SDLRT (r_0=10)	$\mathbf{88.09}_{\pm 0.73}$	$\mathbf{75.08}_{\pm 1.37}$	$83.70_{\pm 0.32}$	2.06M (0.66×)
DLRT (r_0=10)	$87.36_{\pm 0.63}$	$74.92_{\pm 0.41}$	$83.74_{\pm 0.22}$	2.06M (0.66×)
LoRA (r=10)	$87.60_{\pm 1.05}$	$74.98_{\pm 1.55}$	$\mathbf{83.90}_{\pm 0.58}$	3.13M (1.00×)

SDLRT scales cleanly to the 3B regime: it attains the best accuracy on RTE and WiC and is statistically on par with the baselines on BoolQ, while deploying an adapter only $0.66\times$ the size of LoRA's. This confirms that the compensation mechanism remains effective at billion-parameter scale rather than being specific to the smaller encoder of the main experiments.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims are supported by corresponding sections in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 explicitly acknowledges that the negative-feedback factor ω is set heuristically and that scaling SDLRT to large architectures remains future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Theorem 3.1 is derived in detail in Section 3.2. Theorem 3.2 is proved in Appendix B and Theorem 3.3 is proved in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 4 specifies all key hyperparameters, model architectures and datasets.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is given in the supplementary material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 reports the optimizer (SGD with momentum), learning rate schedule, batch size, number of epochs, tolerance τ, and feedback factor ω. Section 4.3 provides the fine-tuning configuration including learning rate, warm-up schedule, weight decay, and per-task epoch counts. All experiments use standard train/test splits of the respective benchmarks.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All image-classification experiments (Figures 3 and 4a) report min/max error bars over 5 independent seeds. Table 2 reports mean $\pm$ standard deviation over 6 seeds. The source of variability (random initialization seed) is stated explicitly.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide an overview of the used compute resources in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work proposes an optimization algorithm for neural network compression. It does not involve human subjects, personally identifiable information, or any application with direct potential for harm. We have reviewed the NeurIPS Code of Ethics and believe our research conforms with it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please see the broader impact statement in Appendix A

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release pre-trained models, datasets, or any artifacts that pose a high risk for misuse. The contribution is a training algorithm evaluated on publicly available benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the following existing assets: CIFAR-10 and CIFAR-100 (MIT License); SuperGLUE benchmark (Creative Commons, per individual dataset licenses); DeBERTa-v3 (MIT License, available on HuggingFace).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.