LEARNING TO BE TRUNCATED: PREPARING LLM CHECKPOINTS FOR SVD COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Compression based on singular value decomposition (SVD) often requires subsequent fine-tuning or weight correction to mitigate performance degradation. Budget-specific optimization must be repeated for different compression ratios. We introduce LLift, which prepares a pretrained model for SVD compression across ratios without further fine-tuning. LLift fine-tunes a low-rank approximation at a single ratio using the original model as a teacher. We merge the learned updates into the original weights while preserving the complementary components excluded from the training forward pass. We apply SVD to the prepared model's target matrices and reuse the decompositions at different truncation ranks. On LLaMA-7B, we demonstrate accuracy gains across three compression ratios using a single prepared model. The gains extend across model families, model sizes, and SVD compression operators. We analyze gradients and an early training update to provide local evidence that updates learned at one ratio can benefit compression at other ratios. These results show that LLift can improve compression across deployment budgets without fine-tuning each compressed model separately.

1 INTRODUCTION

Deploying large language models across hardware with different memory, latency, and compute budgets motivates compression methods that produce models of different sizes from a single pretrained checkpoint. Singular value decomposition (SVD) supports post hoc compression by replacing weight matrices with low-rank factors of adjustable rank. These models use standard dense matrix operations (Wang et al., 2025b). However, truncation can degrade the model's predictions. Existing methods reduce this degradation through activation-aware reconstruction, rank allocation, and error correction (Yuan et al., 2023; Wang et al., 2025a;b; Hu et al., 2026; Abbasi et al., 2026).

Figure 1 compares different methods for producing compressed models at multiple compression ratios. Post-compression fine-tuning and budget-specific reconstruction improve individual compressed models, but require repeating optimization for each target. Elastic methods such as FlexRank and ACIP amortize training across budgets using shared low-rank factors or a learned component ordering (Zaccone et al., 2026; Genzel et al., 2025). These methods instantiate deployment models from the learned representations. We study a complementary goal: updating the original model, then truncating shared SVDs of the prepared model's target matrices at different ranks.

Training with future compression in mind provides a foundation for this goal. Compression-aware training encourages low-rank matrices (Alvarez & Salzmann, 2017), while DecomposableNet jointly trains full- and low-rank networks (Yaguchi et al., 2021). More recently, Prehab prepares the original model by optimizing a task loss together with spectral regularization (Qin et al., 2025), encouraging weight spectra that are more favorable to truncation. Our approach instead uses prediction errors exposed by a low-rank model to guide updates to the original model.

We introduce LLift, which fine-tunes a low-rank approximation of the original model at one preparation budget. This approximation is the low-rank view: its forward pass uses only the retained components, while the original model provides supervision. After fine-tuning, we merge the learned update into the original weights and preserve the components excluded from the training forward pass. With fixed activation transforms, we apply SVD once per target matrix of the prepared model and select different truncation ranks from the shared decompositions. This separates the budget used to expose compression error during training from the budget selected for deployment.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

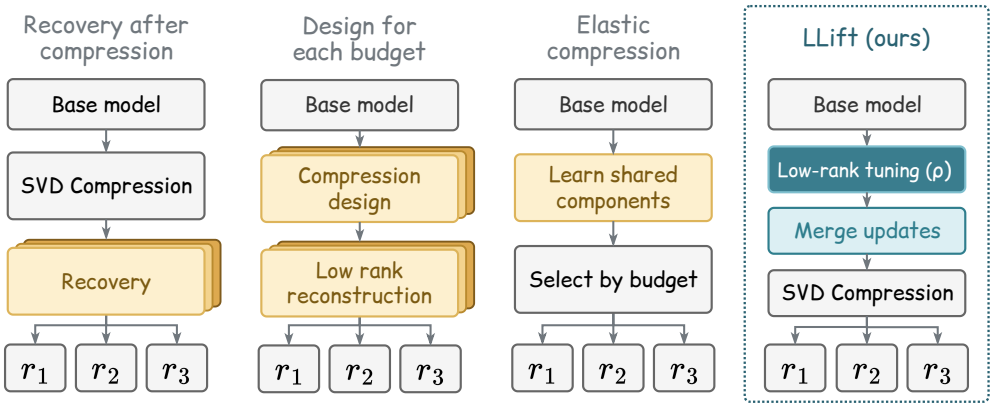


Figure 1: Compression workflows across budgets. Post-compression fine-tuning and budget-specific reconstruction repeat optimization for each budget, while elastic methods select from learned shared components. LLift fine-tunes a low-rank approximation at one preparation budget ρ and merges the learned updates into the original weights to prepare one model for compression across budgets. Stacked boxes indicate budget-specific work; r_1, r_2, r_3 denote deployment budgets.

The challenge is to make updates learned at one budget useful when we compress the prepared model at another. Exporting only the fine-tuned student would discard components absent from the training view, including information that larger deployment budgets could retain. LLift preserves these complementary components while incorporating the learned correction into the original weights. Recompressing the prepared model can then draw on information outside the preparation subspace.

We show that preparing each model at one budget improves compression quality at other evaluated budgets, with gains across model families, sizes, and SVD operators. We analyze gradients at initialization and identify shared descent directions as local evidence of cross-budget compatibility.

Our contributions are threefold:

- We introduce LLift, a model-preparation method that uses supervision at one compression budget. We truncate shared SVDs of the prepared model's target matrices to obtain models with competitive performance across deployment budgets without further fine-tuning.
- We show that updates learned by the low-rank student remain useful after merging the updates and applying SVD to the prepared model. Preserving the original complementary weights improves recompressed accuracy over exporting the fine-tuned student alone at deployment.
- We evaluate model preparation across model families, sizes, budgets, and SVD compression operators, and analyze local cross-budget compatibility through gradient alignment at initialization and recompression after the first nonzero AdamW update to the original model.

2 Related Work

SVD-based LLM compression. Post hoc SVD compression has evolved from weight-space decomposition toward activation-aware reconstruction, rank selection, and error correction. Existing methods use activation-derived scaling (Yuan et al., 2023), input-space whitening (Wang et al., 2025b), differentiable rank optimization (Wang et al., 2025a; Sundrani et al., 2025), cumulative-error suppression (Hu et al., 2026), and loss-aware singular-component selection (Abbasi et al., 2026). Low-Rank Prehab encourages a more compressible weight spectrum before applying SVD (Qin et al., 2025), while NuMuon promotes low-rank structure during pretraining through constrained optimizer updates (Dolatabadi et al., 2026). PiSSA (Meng et al., 2024) adapts principal singular components for parameter-efficient fine-tuning, retaining a frozen residual in its forward pass. LLift prepares a pretrained model by supervising its end-to-end predictions through an activation-aware low-rank view. Its objective targets compressed-model behavior without explicitly regularizing the weight spectrum, and we compress the prepared model using SVD-LLM or ZS-SVD at deployment.

Multi-budget and elastic low-rank deployment. Nested and elastic models support multiple deployment budgets through shared representations or subnetworks, with smaller configurations contained in larger ones (Kusupati et al., 2022; Devvrit et al., 2024; Cai et al., 2024; Taghibakhshi et al., 2025). For low-rank deployment, selecting subsets of shared components adjusts model size. FlexRank combines shared low-rank factors with budget-specific rank configurations and trains the resulting nested models across budgets (Zaccone et al., 2026). ACIP learns a global importance ordering over SVD components together with a shared low-rank correction, allowing deployment at different model sizes by truncating the ordering (Genzel et al., 2025). Both methods organize multi-budget deployment around a shared low-rank representation or component ordering and use these representations to instantiate deployment models directly at different budgets.

LLift instead prepares the original model through supervision of a low-rank view at a single budget and merges the learned updates into the original weights. At deployment, we obtain different model sizes by selecting truncation ranks from shared SVDs of the prepared model's target matrices. We test whether this preparation improves performance at other compression budgets.

3 PRELIMINARIES

Activation-aware low-rank approximation. For a linear layer with weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and calibration activations $X \in \mathbb{R}^{d_{\text{in}} \times N}$, activation-aware compression seeks a rank-k weight approximation that best preserves the original layer outputs on these calibration activations:

$$\min_{\text{rank}(\widehat{W}) \le k} \left\| (W - \widehat{W})X \right\|_F^2 . \tag{1}$$

Following the activation-whitening formulation of SVD-LLM (Wang et al., 2025b), we use $XX^\top$ to summarize the calibration activations' second-order statistics. Adding ϵI, with $\epsilon > 0$, makes this matrix positive definite. We can therefore obtain an invertible factor T satisfying $TT^\top = XX^\top + \epsilon I$. We can compute T by Cholesky decomposition and obtain the following identity:

$$\left\| (W - \widehat{W})T \right\|_F^2 = \left\| (W - \widehat{W})X \right\|_F^2 + \epsilon \left\| W - \widehat{W} \right\|_F^2 . \tag{2}$$

Approximating WT minimizes the output reconstruction error together with a weight reconstruction penalty. We decompose $WT = U\Sigma V^\top$, with singular values in descending order. Since T^{-1} preserves rank, we solve the damped objective by truncating WT and multiplying by T^{-1}:

$$\widehat{W}^{(k)} = U_{:k}\Sigma_{:k}V_{:k}^\top T^{-1}, \tag{3}$$

Here $U_{:k}$ and $V_{:k}$ contain the first k columns of U and V, respectively. We retain the leading $k \times k$ diagonal block of Σ in $\Sigma_{:k}$, corresponding to the k largest singular values of WT.

Nested rank truncation. For a fixed checkpoint and activation transform, the components retained at rank k_1 are a subset of those retained at rank k_2 whenever $k_1 < k_2$. A single SVD can therefore produce multiple model sizes by retaining different numbers of leading components. This nesting permits reuse of the decomposition but does not by itself guarantee task performance at unseen budgets. LLift uses activation-aware SVD to initialize its low-rank training view. At deployment, we apply SVD to the prepared model's transformed target matrices and select different truncation ranks from these shared decompositions to obtain models at each deployment budget.

4 METHOD

LLift prepares a model for SVD compression across deployment budgets. We fine-tune a low-rank student at one preparation anchor, using fixed subspaces derived from the original weights and activation statistics. We merge these updates into the original weights, preserving the complementary components excluded from the student forward pass. For deployment, we truncate shared SVDs of the prepared model's transformed target matrices at the ranks specified by each budget (Figure 2).

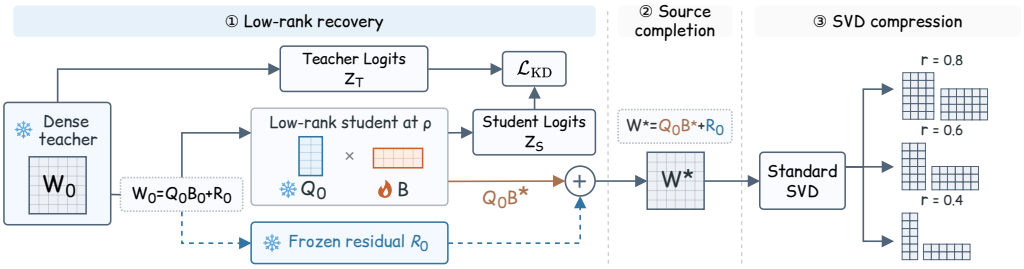


Figure 2: Fine-tuning a low-rank view and merging its updates into the original model. LLift uses the frozen original model to supervise a student whose target layer uses Q_0B at preparation anchor ρ, with Q_0 fixed and B trainable. Snowflakes and flames denote frozen and trainable quantities. After training, we merge the update $Q_0(B^\star - B_0)$ into W_0, yielding $W^\star = Q_0B^\star + R_0$, where $R_0 = W_0 - Q_0B_0$ is the frozen complementary component that bypasses the training forward pass. With a fixed activation transform T, we decompose $W^\star T$ once and truncate the shared decomposition to the ranks specified by r. The diagram shows one target matrix; the prepared model also retains the embeddings, output head, and normalization parameters adapted during student fine-tuning.

4.1 Learning through a Low-Rank View

Activation-aware SVD constructs each compressed layer from local weight and activation statistics, while the quality of the resulting model depends on the joint effect of compression across layers. LLift therefore exposes all target layers to their low-rank views simultaneously and uses the pretrained model to supervise the resulting end-to-end behavior. We use one preparation anchor to define the training view and select deployment ranks when compressing the prepared model.

Let $f(x;\Theta_0)$ denote the original model's logits for input x, with target matrices $\{W_{\ell,0}\}_{\ell\in\mathcal{S}}$ and remaining parameters Φ_0. We use ρ for preparation and r for deployment; both specify nominal retained-parameter fractions in the target matrices. We assign ρ uniformly across target matrices. A rank-k_ℓ factorization stores $k_\ell(d_{\text{out},\ell}+d_{\text{in},\ell})$ parameters instead of $d_{\text{out},\ell}d_{\text{in},\ell}$, so we choose

$$k_\ell(\rho) = \max\left\{1, \left\lfloor \frac{\rho\, d_{\text{out},\ell}d_{\text{in},\ell}}{d_{\text{out},\ell}+d_{\text{in},\ell}} \right\rfloor\right\}. \tag{4}$$

The rank $k_\ell(\rho)$ determines the dimensions of the two factors used in the training forward pass. Uniform parameter retention, subject to rank rounding, does not imply the same retained fraction of singular directions across target matrices with different input and output dimensions.

The initial view preserves the leading activation-aware components of the pretrained weights. We estimate T_ℓ^{train} from the original model on calibration text and decompose $W_{\ell,0}T_\ell^{\text{train}} = U_\ell\Sigma_\ell V_\ell^\top$ as in Eq. 3. We use the retained left singular vectors as a fixed basis and project the weights:

$$Q_{\ell,0} = U_{\ell,:k_\ell(\rho)}, \qquad B_{\ell,0} = Q_{\ell,0}^\top W_{\ell,0}. \tag{5}$$

Here $Q_{\ell,0}\in\mathbb{R}^{d_{\text{out},\ell}\times k_\ell(\rho)}$ and $B_{\ell,0}\in\mathbb{R}^{k_\ell(\rho)\times d_{\text{in},\ell}}$. The product $Q_{\ell,0}B_{\ell,0} = Q_{\ell,0}Q_{\ell,0}^\top W_{\ell,0}$ is the rank-$k_\ell(\rho)$ projection induced by the activation-aware decomposition in Eq. 3. We associate each coordinate matrix B_ℓ with a corresponding model weight parameterized as

$$W_\ell(B_\ell) = W_{\ell,0} + Q_{\ell,0}(B_\ell - B_{\ell,0}).$$

Since $Q_{\ell,0}^\top Q_{\ell,0} = I$, its projection onto the fixed column space of $Q_{\ell,0}$ satisfies $Q_{\ell,0}Q_{\ell,0}^\top W_\ell(B_\ell) = Q_{\ell,0}B_\ell$. The student evaluates only the projected map $h \mapsto Q_{\ell,0}(B_\ell h)$; the complementary component of the original weight remains unchanged and does not participate in this forward pass. Preparation therefore does not require optimizing full target matrices. We keep $Q_{\ell,0}$ fixed.

The original model serves as a frozen teacher. Let $z_t^{\text{T}} = f(x;\Theta_0)_t$ denote its logits at position t and $z_t^{\text{S}}(\mathcal{B},\Phi)$ those of the student with target weights $\{Q_{\ell,0}B_\ell\}_{\ell\in\mathcal{S}}$, where $\mathcal{B} = \{B_\ell\}_{\ell\in\mathcal{S}}$ and Φ collects other student parameters. We minimize the token-level student-to-teacher KL loss:

$$\mathcal{L}_{\text{LLift}}(\mathcal{B},\Phi) = \mathbb{E}_{x\sim\mathcal{D}_{\text{train}}}\left[\frac{1}{|\Omega(x)|}\sum_{t\in\Omega(x)} \text{KL}\big(p_\tau(z_t^{\text{S}}(\mathcal{B},\Phi))\,\big\|\,p_\tau(z_t^{\text{T}})\big)\right], \tag{6}$$

where $p_\tau(z) = \mathrm{softmax}(z/\tau)$ and $\Omega(x)$ excludes padded positions. Supervision at the final logits accounts for the combined effect of compressing all target layers. We initialize $(\mathcal{B}, \Phi) \leftarrow (\mathcal{B}_0, \Phi_0)$ and jointly optimize these parameters. We train the embeddings, output head, and normalization parameters while keeping the teacher and the bases $Q_{\ell,0}$ fixed throughout preparation.

4.2 Constructing the Prepared Model

We update the original weights by merging the coordinate change $B_\ell^\star - B_{\ell,0}$ into $W_{\ell,0}$ through $Q_{\ell,0}$, preserving the original components that the training forward pass excludes. We can equivalently combine the fine-tuned low-rank factors with the original truncation residual.

We retain the original truncation residual

$$R_{\ell,0} = W_{\ell,0} - Q_{\ell,0}B_{\ell,0}, \qquad Q_{\ell,0}^\top R_{\ell,0} = 0. \tag{7}$$

We write each target weight in the prepared model in two equivalent forms:

$$\begin{aligned} W_\ell^\star &= W_{\ell,0} + Q_{\ell,0}(B_\ell^\star - B_{\ell,0}) \\ &= Q_{\ell,0}B_\ell^\star + R_{\ell,0}. \end{aligned} \tag{8}$$

Reattaching $R_{\ell,0}$ yields the minimum-change completion of the fine-tuned coordinates:

$$W_\ell^\star = \underset{W:\, Q_{\ell,0}^\top W = B_\ell^\star}{\arg\min} \|W - W_{\ell,0}\|_F^2. \tag{9}$$

The constraint fixes the trained component, and the nearest completion preserves the original complement. We prove this minimum-distance property in Appendix A.1; it does not establish task-loss optimality. With no adaptation, Eq. 8 recovers $W_{\ell,0}$. After adaptation, projection onto the fixed basis still recovers the trained view: $Q_{\ell,0}Q_{\ell,0}^\top W_\ell^\star = Q_{\ell,0}B_\ell^\star$.

The fixed $R_{\ell,0}$ requires no gradients or optimizer states. We reattach it only when exporting the prepared model. The target matrices require optimizer states only for B_ℓ. The exported model is $\Theta^\star = (\{W_\ell^\star\}_{\ell\in\mathcal{S}}, \Phi^\star)$, including the adapted remaining parameters. In Appendix A, we derive when factorized and full-matrix parameterizations induce identical updates to the target weights.

4.3 Recompression beyond the Training Budget

The intended benefit of preparation is improved quality after recompressing the prepared model, including at budgets not seen during preparation. Deployment compression acts on the completed weights $W_\ell^\star = Q_{\ell,0}B_\ell^\star + R_{\ell,0}$, so both the learned update and the preserved complementary component can influence the singular directions retained by SVD at each deployment budget.

The orthogonality condition $Q_{\ell,0}^\top R_{\ell,0} = 0$ refers to the fixed preparation basis. For a deployment activation transform T, it does not imply $B_\ell^\star T T^\top R_{\ell,0}^\top = 0$. The cross terms in $(W_\ell^\star T)(W_\ell^\star T)^\top$ can therefore couple the fine-tuned component to the preserved complement. Applying SVD to $W_\ell^\star T$ can select left singular directions that differ from the preparation basis, allowing the complement to affect compression even when $r \le \rho$. At $r = \rho$, the recompressed model consequently need not reproduce the predictions of the low-rank student trained at that same preparation budget.

We hold the deployment activation transforms T_ℓ^{deploy} fixed across budgets and apply SVD once to each $W_\ell^\star T_\ell^{\mathrm{deploy}}$. For budget r, we retain the first $k_\ell(r)$ components of these shared decompositions and construct the layer factors as in Eq. 3. Let $\mathcal{C}_r(\Theta; \mathcal{D}_{\mathrm{cal}}^{\mathrm{deploy}})$ denote compression with these fixed transforms. Applying this operator to the prepared model yields the deployment models

$$\Theta_r = \mathcal{C}_r\left(\Theta^\star; \mathcal{D}_{\mathrm{cal}}^{\mathrm{deploy}}\right), \qquad r \in \mathcal{R}. \tag{10}$$

Each budget requires its own truncation ranks and factor materialization, while reusing the SVD of each target matrix without additional fine-tuning. Section 5.3 examines cross-budget transfer through the alignment of the ratio-specific fine-tuning gradients at initialization.

5 Experiments

We test whether preparation at one anchor improves compression across budgets, examine residual restoration, and assess generality across SVD operators, model families, and scales.

Table 1: SVD compression results on LLaMA-7B. Bold and underlined WT2, C4, and Avg. values mark the best and second-best results at each deployment budget. †: budget-specific optimization.

r	Method	WT2↓	C4↓	OBQA↑	ARC-e↑	ARC-c↑	WinoG↑	HellaS↑	PIQA↑	MathQA↑	Avg.↑
Uncompressed		5.68	7.34	0.31	0.75	0.42	0.69	0.56	0.78	0.26	0.54
0.8	ASVD	11.14	15.93	0.25	0.53	0.27	0.64	0.41	0.68	0.24	0.43
	SVD-LLM	7.94	15.84	0.22	0.58	0.29	0.63	0.43	0.69	0.24	0.44
	Dobi-SVD†	8.54	**10.01**	0.26	0.59	0.31	0.66	0.44	0.70	0.23	0.46
	SAES-SVD†	7.17	13.77	0.29	0.68	0.36	0.65	0.45	0.75	0.25	0.49
	ZS-SVD	<u>6.74</u>	10.74	0.31	0.70	0.37	0.68	0.49	0.74	0.24	<u>0.50</u>
	ZS-SVD 5×†	**6.43**	<u>10.42</u>	0.32	0.72	0.39	0.67	0.50	0.75	0.25	**0.51**
	LLift	7.49	11.15	0.30	0.67	0.37	0.66	0.52	0.74	0.25	<u>0.50</u>
0.6	ASVD	1407	1109	0.13	0.28	0.22	0.48	0.26	0.55	0.19	0.30
	SVD-LLM	13.11	49.83	0.19	0.42	0.25	0.58	0.33	0.60	0.21	0.37
	Dobi-SVD†	13.54	<u>23.54</u>	0.22	0.41	0.27	0.58	0.34	0.61	0.23	0.38
	SAES-SVD†	10.42	32.79	0.23	0.50	0.29	0.62	0.36	0.65	0.23	0.41
	ZS-SVD	11.44	34.13	0.23	0.52	0.26	0.62	0.35	0.64	0.22	0.41
	ZS-SVD 5×†	<u>9.45</u>	26.20	0.23	0.56	0.29	0.62	0.38	0.66	0.22	<u>0.42</u>
	LLift	**8.40**	**14.77**	0.25	0.63	0.31	0.62	0.45	0.71	0.25	**0.46**
0.4	ASVD	57057	43036	0.12	0.26	0.21	0.49	0.26	0.53	0.18	0.29
	SVD-LLM	53.74	345.49	0.14	0.28	0.22	0.50	0.27	0.55	0.21	0.31
	Dobi-SVD†	46.18	190.62	0.15	0.31	0.20	0.52	0.28	0.54	0.22	0.32
	SAES-SVD†	22.01	93.97	0.16	0.33	0.25	0.52	0.30	0.54	0.23	<u>0.34</u>
	ZS-SVD	45.17	212.57	0.15	0.31	0.21	0.54	0.28	0.55	0.21	0.32
	ZS-SVD 5×†	<u>20.41</u>	<u>88.56</u>	0.17	0.37	0.22	0.55	0.30	0.57	0.21	<u>0.34</u>
	LLift	**13.48**	**32.24**	0.17	0.48	0.24	0.57	0.34	0.62	0.22	**0.38**

5.1 Experimental Setup

Preparation. LLaMA-7B is our reference model. Unless otherwise stated, we prepare each model once at $\rho = 0.6$ on the WikiText-2 (WT2) (Merity et al., 2017) training split using KL distillation from the original model. We construct the fixed preparation bases from original-model activations on WT2. We use student-to-teacher KL as the default objective and report CE and KL+CE controls in Appendix B.2. In particular, removing KL supervision and training with CE alone substantially increases post-compression perplexity, especially at smaller deployment budgets.

Deployment. We use SVD-LLM to obtain deployment models at $r \in \{0.8, 0.6, 0.4\}$ without further fine-tuning. For matched comparisons, we estimate deployment activation transforms from the original model on a separate WT2/Alpaca (Taori et al., 2023) calibration set and hold them fixed across models and budgets. We reuse each model's SVDs across all three budgets by varying the truncation ranks. Re-estimating these transforms from the prepared model changes accuracy only modestly and provides no consistent perplexity benefit (Appendix B.3). Following Eq. 4, r denotes the nominal ratio of factor parameters to original target-matrix parameters.

Evaluation. We report held-out WT2 and C4 (Raffel et al., 2020) perplexity, together with the equally weighted mean of plain zero-shot accuracy (Avg.) on seven tasks: OpenBookQA, ARC-Easy, ARC-Challenge, WinoGrande, HellaSwag, PIQA, and MathQA (Mihaylov et al., 2018; Clark et al., 2018; Sakaguchi et al., 2019; Zellers et al., 2019; Bisk et al., 2019; Amini et al., 2019).

5.2 Quality Across Deployment Budgets

Comparison with SVD compression methods. Table 1 compares LLift with ASVD (Yuan et al., 2023), SVD-LLM (Wang et al., 2025b), Dobi-SVD (Wang et al., 2025a), SAES-SVD (Hu et al., 2026), and ZS-SVD (Abbasi et al., 2026). LLift raises mean accuracy from the reported SVD-LLM scores of $0.44/0.37/0.31$ to $0.50/0.46/0.38$ at $r = 0.8/0.6/0.4$. At $r = 0.6/0.4$, it achieves the highest mean accuracy and lowest perplexity on both corpora among the methods in the table. At these two budgets, accuracy increases from $0.42/0.34$ with ZS-SVD $5\times$ to $0.46/0.38$ with LLift.

Table 2: Elastic methods. Seven-task average accuracy (↑). Δ is a rank-32 correction whose parameters are excluded from r.

Method	$r = 0.8$	$r = 0.6$	$r = 0.4$
ACIP	0.49	0.40	0.33
ACIP ($+\Delta$)	0.48	0.45	0.38
FlexRank	0.49	0.46	0.41
LLift	0.50	0.46	0.38

Table 3: Model preparation and residual restoration under matched `mix75` calibration. Seven-task average accuracy (↑).

Model	$r = 0.8$	$r = 0.6$	$r = 0.4$
Original	0.44	0.40	0.31
Spectral regularization	0.48	0.41	0.33
Fine-tuned $Q_0B^\star$ only	0.44	0.44	0.37
LLift: $Q_0B^\star + R_0$	0.50	0.46	0.38

Comparison with elastic methods. Table 2 compares LLift, FlexRank, and ACIP with and without its rank-32 correction Δ. ACIP without Δ obtains $0.49/0.40/0.33$ accuracy at $r = 0.8/0.6/0.4$, versus $0.50/0.46/0.38$ for LLift. FlexRank obtains accuracies of $0.49/0.46/0.41$ at these budgets, matching LLift in mean accuracy at $r = 0.6$ and outperforming it at $r = 0.4$.

Budget-specific fine-tuning. At $r = 0.8/0.6/0.4$, SVD-LLM followed by two-stage LoRA fine-tuning on Alpaca achieves $0.50/0.46/0.39$ accuracy, compared with $0.50/0.46/0.38$ for LLift using one preparation run (Appendix B.6). The three fine-tuning runs take about 10.8 GPU-hours (3.6 per budget); LLift's optimization and basis construction take 1.65 A100 GPU-hours once (Appendix C).

5.3 Model Preparation and Cross-Budget Transfer

Relation to Low-Rank Prehab. Low-Rank Prehab (Qin et al., 2025) prepares a model before SVD, optimizing a task loss on the original model with spectral rank regularization. LLift supervises a low-rank view at one anchor and restores the original complement. We report our Prehab reimplementation's setup and results in Appendix B.7. A separate control using truncated Schatten spectral regularization improves mean accuracy over Original but remains below LLift at all three budgets (Table 3). We detail the control's objective and training setup in Appendix B.7.

Model preparation and residual restoration. In Table 3, LLift raises mean accuracy from $0.44/0.40/0.31$ to $0.50/0.46/0.38$ at $r = 0.8/0.6/0.4$. A single prepared model therefore benefits deployment budgets both above and below the preparation anchor.

To isolate residual restoration, we apply SVD-LLM to the fine-tuned student $Q_0B^\star$ and the prepared model $Q_0B^\star + R_0$ from the same preparation run, with identical adapted non-target parameters. Restoring R_0 improves recompressed accuracy from 0.44 to 0.50 at $r = 0.8$, and from $0.44/0.37$ to $0.46/0.38$ at $r = 0.6/0.4$. The largest gain occurs above the preparation anchor, where restoring the original complement makes directions outside the preparation subspace available to recompression. Gains at and below the anchor indicate benefits beyond larger deployment ranks, consistent with the restored complement influencing the subspace selected by SVD (Section 4.3).

Role of non-target parameters. We freeze LLaMA-7B's embeddings, output head, and normalization parameters and train only the factor coordinates. Both variants' mean accuracies round to $0.50/0.46/0.38$, whereas freezing these parameters increases WT2 perplexity by 12–13% and C4 perplexity by approximately 4–5% across the three deployment budgets (Appendix B.1). We find that adapting these parameters primarily improves perplexity in this control.

Spectral energy retention. The accuracy gains occur with nearly unchanged retained whitened spectral energy. Across the three deployment budgets, the reported retained-energy fractions of the original and prepared models differ by at most 2×10^{-4} (Appendix B.4). This suggests that retained energy alone does not fully characterize the quality of models after SVD compression.

Preparation anchor. Preparing at $\rho = 0.4$ improves accuracy at $r = 0.6/0.4$ relative to the default configuration (Figure 3(a)). The default $\rho = 0.6$ yields the lowest WT2 perplexity at all three deployment budgets among the tested anchors (Appendix B.2), revealing a tradeoff between task accuracy and language-modeling quality. The anchor-matched points ($\rho = r$) in Figure 3(a) provide budget-specific preparation references. The single default $\rho = 0.6$ prepared model remains competitive, with accuracy differences of $+0.50/0/-1.66$ percentage points at $r = 0.8/0.6/0.4$.

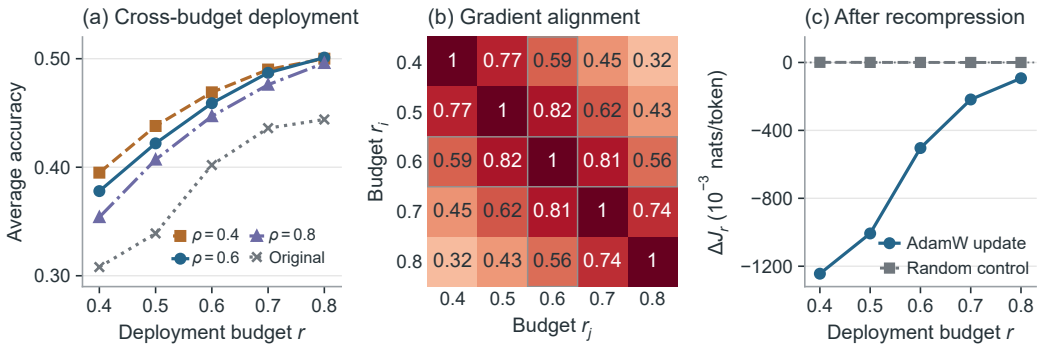


Figure 3: Cross-budget transfer and local compatibility on LLaMA-7B. (a) Seven-task mean accuracy across deployment budgets for models prepared at different anchors; we include the original model as a reference. (b) Cosine similarities of fixed-projector teacher-KL gradients at initialization. (c) Change in held-out student-to-teacher KL after updating the original model and recompressing it with SVD. We compare the first nonzero AdamW update ($s = 1$) with a norm-matched random update. Negative values indicate lower KL than when we compress the original model.

Figure 3(a) further evaluates intermediate deployment budgets $r = 0.5$ and 0.7, which we do not use as preparation anchors, providing denser evidence of transfer away from the preparation budget.

Local cross-budget compatibility. We probe teacher-KL objectives at the original checkpoint while holding activation transforms and budget-specific projectors P_r fixed. Let $g_r = \nabla_W \widetilde{\mathcal{L}}_r(W_0; P_r)$, where W collects the sampled target matrices, and we keep all remaining parameters fixed. For the diagnostic projector P_ρ, an ordinary gradient step in the sampled fixed-basis factor coordinates induces $\Delta W = -\eta g_\rho$ in model weights. For sufficiently small η,

$$\Delta\widetilde{\mathcal{L}}_r = -\eta\langle g_r, g_\rho\rangle_F + O(\eta^2). \tag{11}$$

Positive alignment therefore identifies a shared local descent direction within the sampled parameters. The pairwise mean alignments in Figure 3(b) are positive. At $\rho = 0.6$, cosine similarity is 0.59 ± 0.02 for $r = 0.4$ and 0.56 ± 0.11 for $r = 0.8$, with standard deviations across five disjoint batches. This diagnostic tests local compatibility at initialization; the model comparisons evaluate transfer after AdamW preparation and SVD compression of the prepared model. Appendix B.5 provides the factor-coordinate derivation and probe details. This fixed-projector diagnostic does not test whether the same local benefit survives an AdamW update followed by SVD of the updated weights; the following probe evaluates the combined effect of these two operations.

Realized update after recompression. We scale the first nonzero AdamW update from the default $\rho = 0.6$ run by s and apply it to the original model, then recompute the deployment SVD factors from the updated weights. Let ΔJ_r denote the change in held-out WT2 student-to-teacher KL relative to compressing the original model at the same budget r. KL decreases at all five tested budgets for $s \in \{0.25, 0.5, 1\}$, with ΔJ_r ranging from -1.244 to -0.093 nats/token at $s = 1$ (Figure 3(c)). At $s = 1$, replaying only the target-matrix component gives nearly identical changes, while the norm-matched random control stays near zero. These measurements provide local evidence of cross-budget transfer for the first nonzero update. Appendix B.5 reports the controls.

5.4 Generality Across Compression Operators and Models

Compression with another operator. LLift also improves compression with one-shot ZS-SVD (Table 4). Applying one-shot ZS-SVD to the model prepared with the default SVD-LLM basis raises mean accuracy from $0.42/0.33$ to $0.45/0.38$ at $r = 0.6/0.4$. At $r = 0.8$, both the original and prepared models achieve approximately 0.51 accuracy. C4 perplexity improves at all three budgets, while WT2 perplexity improves at $r = 0.6/0.4$ but increases at $r = 0.8$. The gains at smaller budgets show that preparation using an SVD-LLM basis also benefits ZS-SVD compression.

Table 4: ZS-SVD on LLaMA-7B. Rows 1–3 use one-shot compression with matched `mix75` calibration; parentheses identify preparation bases. Abbasi et al. (2026) report ZS-SVD $5\times$ results.

Model / basis	Time (GPU-h)	$r = 0.8$			$r = 0.6$			$r = 0.4$		
		WT2 ↓	C4 ↓	Avg. ↑	WT2 ↓	C4 ↓	Avg. ↑	WT2 ↓	C4 ↓	Avg. ↑
ZS-SVD	0.25	6.86	10.46	0.51	12.41	28.04	0.42	56.71	184.77	0.33
LLift (SVD-LLM)	1.65 + 0.25	7.84	10.11	0.51	9.86	15.38	0.45	16.88	35.33	0.38
LLift (ZS-SVD)	1.65 + 0.25	7.27	10.02	0.51	8.58	14.34	0.46	16.38	42.18	0.36
ZS-SVD $5\times$ (ref)	2.11 / budget	6.43	10.42	0.51	9.45	26.20	0.42	20.41	88.56	0.34

We also prepare LLaMA-7B with a ZS-SVD basis at $\rho = 0.6$, yielding $0.46/0.36$ mean accuracy at $r = 0.6/0.4$, compared with $0.45/0.38$ for the default basis. These results support LLift's robustness to compressor choice across the tested preparation and deployment settings (Appendix B.3).

We record 1.65 A100 GPU-hours for LLaMA-7B optimization and basis construction, paid once per prepared model. The additional 0.25 GPU-hours in Table 4 covers one-shot ZS-SVD at one budget. The iterative ZS-SVD $5\times$ reference takes 2.11 A100 GPU-hours per deployment budget, including five correction rounds. With fixed deployment transforms, we compute one weighted SVD per target matrix of the prepared model and materialize factors separately for each budget (Appendix C).

Table 5: Evaluation across model families. Seven-task average accuracy (↑).

Model	Method	r=0.8	r=0.6	r=0.4
Qwen3-8B	SVD-LLM	0.49	0.35	0.31
	ZS-SVD	0.55	0.41	0.31
	LLift	0.55	0.47	0.34
Gemma-2-9B	SVD-LLM	0.52	0.40	0.32
	ZS-SVD	0.57	0.39	0.32
	LLift	0.57	0.51	0.40

Table 6: Scaling to larger LLaMA models. Seven-task average accuracy (↑).

Model	Method	r=0.8	r=0.6	r=0.4
LLaMA-13B	SVD-LLM	0.50	0.42	0.32
	ZS-SVD	0.54	0.39	0.34
	LLift	0.54	0.49	0.41
LLaMA-30B	SVD-LLM	0.55	0.45	0.33
	ZS-SVD	0.59	0.52	0.38
	LLift	0.57	0.53	0.45

Model families and scales. Tables 5 and 6 extend the evaluation to Qwen3-8B, Gemma-2-9B, LLaMA-13B, and LLaMA-30B. We prepare each model independently and use the same prepared model for all three deployment budgets. At $r = 0.6/0.4$, LLift achieves higher mean accuracy than the SVD-LLM and ZS-SVD references across all four models. At $r = 0.6$, accuracy on Qwen3-8B and Gemma-2-9B rises from $0.35/0.40$ with SVD-LLM to $0.47/0.51$ with LLift. At $r = 0.4$, accuracy reaches 0.41 on LLaMA-13B and 0.45 on LLaMA-30B, compared with 0.32 and 0.33 for SVD-LLM compression. At $r = 0.8$, LLift matches ZS-SVD's seven-task mean accuracy on Qwen3-8B, Gemma-2-9B, and LLaMA-13B, but trails it on LLaMA-30B (0.57 versus 0.59).

6 Conclusion

LLift prepares a reusable model for SVD compression across deployment budgets by fine-tuning a low-rank view at a single preparation budget. The frozen original model supervises the low-rank student. We merge the learned updates into the original weights while preserving components excluded from the student forward pass. We reuse SVDs of the prepared model's transformed target matrices at different truncation ranks to obtain deployment models without further fine-tuning.

On LLaMA-7B, a single prepared model raises seven-task mean accuracy from the reported SVD-LLM scores of $0.44/0.37/0.31$ to $0.50/0.46/0.38$ at $r = 0.8/0.6/0.4$. Controlled comparisons show that preserving the complementary components of the original weights is important for subsequent recompression, while the similar retained spectral energy before and after preparation indicates that spectral energy alone does not explain compression quality. The improvements further extend across model families, model sizes, and SVD-based compression operators. LLift thus shifts fine-tuning from individual compressed models to model preparation, allowing the same prepared model to support multiple deployment budgets without repeating compression-aware optimization.

Reproducibility Statement

We specify LLift's fixed-basis preparation, student-to-teacher KL objective, residual restoration, and deployment compression in Section 4. Appendix A provides the minimum-change completion proof and the analysis of fixed-basis updates under SGD and AdamW. Section 5.1 and Appendix B.7 document the datasets, trainable parameters, training hyperparameters, calibration, evaluation settings, and baseline configurations, including fixed deployment transforms and reuse of each prepared model's SVDs across budgets. Appendix B reports preparation and deployment ablations, together with the retained-energy, gradient-alignment, and AdamW recompression probe protocols. Appendix C records hardware, measured preparation stages, and deployment cost accounting. We will publicly release the LLift implementation, evaluation code, and experiment configurations to support reproduction of our results.

AI Use Statement

In this work, we used generative AI tools to assist with language editing and improving the readability of the manuscript, as well as with limited code development and literature summarization. All AI-assisted content, code, and references were reviewed and verified by the authors. The authors take full responsibility for the final content, claims, and results presented in this work.

References

Ali Abbasi, Chayne Thrash, Haoran Qin, Shansita Sharma, Sepehr Seifi, and Soheil Kolouri. Zero sum SVD: Balancing loss sensitivity for low rank LLM compression, 2026. URL https://arxiv.org/abs/2602.02848.

Jose M. Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/db85e2590b6109813dafa101ceb2faeb-Abstract.html.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model, 2024. URL https://arxiv.org/abs/2406.10260.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. MatFormer: Nested transformer for elastic inference, 2024. URL https://arxiv.org/abs/2310.07707.

Hadi Mohaghegh Dolatabadi, Thalaiyasingam Ajanthan, Sameera Ramasinghe, Chamin P Hewa Koneputugodage, Shamane Siriwardhana, Violetta Shevchenko, Karol Pajak, James Snewin, Gil Avraham, and Alexander Long. NuMuon: Nuclear-norm-constrained Muon for compressible LLM training, 2026. URL https://arxiv.org/abs/2603.03597.

Martin Genzel, Patrick Putzky, Pengfei Zhao, Sebastian Schulze, Mattes Mollenhauer, Robert Seidel, Stefan Dietzel, and Thomas Wollmann. Choose your model size: Any compression of large language models without re-computation, 2025. URL https://arxiv.org/abs/2502.01717.

Xing Hu, Dawei Yang, Yuan Cheng, Zhixuan Chen, and Zukang Xu. SAES-SVD: Self-adaptive suppression of accumulated and local errors for SVD-based LLM compression, 2026. URL https://arxiv.org/abs/2602.03051.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, December 2022.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models, 2024. URL https://arxiv.org/abs/2404.02948v3.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Haoran Qin, Shansita Sharma, Ali Abbasi, Chayne Thrash, and Soheil Kolouri. Low-rank Prehab: Preparing neural networks for SVD compression, 2025. URL https://arxiv.org/abs/2512.01980.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Sidhant Sundrani, Francesco Tudisco, and Pasquale Minervini. Low-rank compression of language models via differentiable rank selection, 2025. URL https://arxiv.org/abs/2512.13733.

Ali Taghibakhshi, Sharath Turuvekere Sreenivas, Saurav Muralidharan, Ruisi Cai, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Yoshi Suhara, Oluwatobi Olabiyi, Daniel Korzekwa, Mostofa Patwary, Mohammad Shoeybi, Jan Kautz, Bryan Catanzaro, Ashwath Aithal, Nima Tajbakhsh, and Pavlo Molchanov. Nemotron Elastic: Towards efficient many-in-one reasoning LLMs, 2025. URL https://arxiv.org/abs/2511.16664.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Qinsi Wang, Jinghan Ke, Masayoshi Tomizuka, Yiran Chen, Kurt Keutzer, and Chenfeng Xu. Dobi-SVD: Differentiable SVD for LLM compression and some new perspectives, 2025a. URL https://arxiv.org/abs/2502.02723.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations (ICLR)*, 2025b. URL https://openreview.net/forum?id=LNYIUouhdt.

Atsushi Yaguchi, Taiji Suzuki, Shuhei Nitta, Yukinobu Sakata, and Akiyuki Tanizawa. Decomposable-Net: Scalable low-rank compression for neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 3249–3256, 2021. doi: 10.24963/ijcai.2021/447. URL https://www.ijcai.org/proceedings/2021/447.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models, 2023.

Riccardo Zaccone, Stefanos Laskaridis, Marco Ciccone, and Samuel Horvath. FlexRank: Nested low-rank knowledge decomposition for adaptive model deployment. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=DK0kvnNelx.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

A Additional Derivations

We prove the minimum-change completion and establish when factorized and full-matrix training produce the same updates.

A.1 Minimum-Change Completion

We omit the layer index and let $Q_0^\top Q_0 = I$, $B_0 = Q_0^\top W_0$, and $R_0 = (I - Q_0 Q_0^\top) W_0$. Every feasible W satisfying $Q_0^\top W = B^\star$ has the unique decomposition $W = Q_0 B^\star + R$ with $Q_0^\top R = 0$. The two terms of $W - W_0 = Q_0(B^\star - B_0) + (R - R_0)$ are orthogonal in the Frobenius inner product. Consequently,

$$\|W - W_0\|_F^2 = \|B^\star - B_0\|_F^2 + \|R - R_0\|_F^2. \tag{12}$$

The learned coordinates fix the first term, and $R = R_0$ uniquely minimizes the second. This proves Eq. 9 for each target matrix. The minimization concerns target weights in Frobenius norm; we optimize the remaining model parameters with the training objective in Eq. 6.

A.2 Fixed-Basis Updates under SGD and AdamW

For a fixed orthonormal basis Q and projector $P = QQ^\top$, write

$$W = QB + R, \qquad B = Q^\top W, \qquad R = (I - P)W. \tag{13}$$

The full-matrix forward PW equals the factorized forward QB. For upstream gradient $G = \partial \mathcal{L}/\partial(PW)$, their gradients satisfy $\nabla_W \mathcal{L} = QQ^\top G$ and $\nabla_B \mathcal{L} = Q^\top G$. Under plain SGD without momentum or weight decay, an update to the full matrix therefore gives

$$W_{t+1} = W_t - \eta_t QQ^\top G_t = Q\big(B_t - \eta_t Q^\top G_t\big) + R_0. \tag{14}$$

With matched initialization, minibatches, learning rates, and updates to other parameters, the two implementations produce the same student trajectory and the same prepared model with target weights $QB_* + R_0$. For this fixed-basis SGD setting, factor coordinates therefore suffice to represent the full-matrix training trajectory.

This equivalence does not generally extend to AdamW. Its elementwise second moment statistics and adaptive scaling depend on whether the optimizer acts in the coordinates of W or B. Even when the gradient with respect to W lies in the column space of Q, the adaptively scaled direction A_t need not do so. For an AdamW update to W with decay coefficient λ,

$$W_{t+1} = (1 - \eta_t \lambda) W_t - \eta_t A_t, \qquad R_{t+1} = (1 - \eta_t \lambda) R_t - \eta_t (I - P) A_t. \tag{15}$$

Thus full-matrix AdamW can update the complementary component, whereas factorized training followed by residual restoration preserves R_0. Refreshing the projector additionally changes the training subspace.

B Additional LLift Results and Evaluation Details

We examine preparation choices, compression operators, and cross-budget diagnostics. We report accuracy on a 0–1 scale and calculate differences before rounding scores to two decimal places.

B.1 Freezing the Remaining Parameters

To isolate the contribution of Φ, we freeze the token embeddings, output head, and all 65 RMSNorm modules in LLaMA-7B and train only the factor coordinates $\mathcal{B}$. We keep the KL objective, anchor $\rho = 0.6$, and other preparation and deployment settings unchanged.

Table 7 shows that adapting Φ primarily benefits post-compression perplexity. Both variants yield the same mean accuracies after rounding to two decimals. The unrounded mean accuracies differ by at most 0.0022 on the 0–1 accuracy scale. Freezing these parameters increases WT2 perplexity by 12–13% and C4 perplexity by approximately 4–5% across the three budgets.

Table 7: Effect of freezing the non-target parameters Φ on compressed LLaMA-7B models. Both variants train the factor coordinates with KL at $\rho = 0.6$ and restore the original residuals before compression.

	WT2 PPL ↓		C4 PPL ↓		Avg. acc. ↑	
Deployment	Trainable	Frozen	Trainable	Frozen	Trainable	Frozen
$r = 0.8$	7.49	8.40	11.15	11.58	0.50	0.50
$r = 0.6$	8.40	9.46	14.77	15.46	0.46	0.46
$r = 0.4$	13.48	15.12	32.24	33.65	0.38	0.38

B.2 Objective and Anchor Ablations

We vary the loss while keeping the remaining recipe fixed. KL+CE gives each term weight one; pure CE uses no teacher. Figure 4 shows that CE increases C4 perplexity at all three budgets, from 32.2 to 90.6 at $r = 0.4$. Its accuracy slightly exceeds KL at $r = 0.6$ but falls to 0.37 at $r = 0.4$, versus 0.38 for KL. KL+CE improves mean accuracy over KL by 0.10–0.83 percentage points while slightly increasing C4 perplexity at every budget. We use KL to balance task accuracy and language-modeling quality.

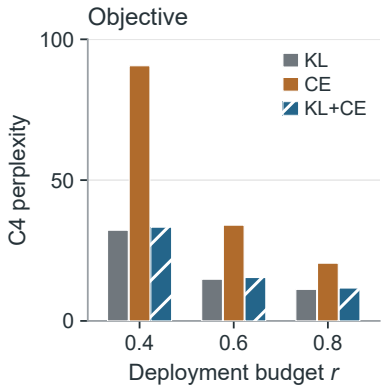


Figure 4: Training-objective ablation on LLaMA-7B. C4 perplexity for KL, CE, and equally weighted KL+CE at each deployment budget, with preparation anchor $\rho = 0.6$.

Table 8 complements the anchor accuracy comparison in Figure 3(a) with WT2 perplexity. Among the tested anchors, $\rho = 0.6$ yields the lowest perplexity at each budget.

Table 8: Preparation-anchor ablation on LLaMA-7B. WT2 perplexity (↓) after SVD-LLM compression. Each KL-prepared model is reused at all three deployment budgets.

Anchor ρ	$r = 0.8$	$r = 0.6$	$r = 0.4$
0.4	9.72	10.89	14.07
0.6 (default)	7.49	8.40	13.48
0.8	7.70	9.53	20.20

B.3 Preparation Bases and Compression Operators

Table 9 groups the preparation-basis comparison by deployment operator. We prepare each LLift model once at $\rho = 0.6$ using either an SVD-LLM or a ZS-SVD basis, then compress it with each operator at three budgets. Both deployment protocols use `mix75` calibration. Each deployment block evaluates the two preparation bases under the same compression operator and evaluation settings.

Table 9: LLaMA-7B results across preparation bases and compression operators under `mix75` calibration. Parentheses identify the preparation basis; both LLift variants use KL at $\rho = 0.6$. WT2 and C4 report perplexity; task columns and Avg. report zero-shot accuracy. Each panel reports one deployment operator.

Preparation	r	WT2	C4	OBQA	ARC-e	ARC-c	WinoG	HellaS	PIQA	MathQA	Avg.
SVD-LLM deployment											
Original	0.8	8.03	13.89	0.23	0.62	0.31	0.63	0.39	0.69	0.23	0.44
	0.6	14.27	35.03	0.24	0.53	0.26	0.57	0.34	0.64	0.22	0.40
	0.4	63.25	218.52	0.15	0.28	0.21	0.49	0.27	0.55	0.22	0.31
LLift (SVD-LLM)	0.8	7.49	11.15	0.30	0.67	0.37	0.66	0.52	0.74	0.25	0.50
	0.6	8.40	14.77	0.25	0.63	0.31	0.62	0.45	0.71	0.25	0.46
	0.4	13.48	32.24	0.17	0.48	0.24	0.57	0.34	0.62	0.22	0.38
LLift (ZS-SVD)	0.8	7.74	11.61	0.27	0.67	0.34	0.66	0.51	0.74	0.26	0.50
	0.6	9.41	17.05	0.24	0.63	0.29	0.63	0.44	0.70	0.24	0.45
	0.4	17.96	52.73	0.17	0.42	0.21	0.55	0.32	0.60	0.22	0.36
ZS-SVD deployment											
Original	0.8	6.86	10.46	0.30	0.71	0.37	0.69	0.50	0.75	0.25	0.51
	0.6	12.41	28.04	0.23	0.59	0.29	0.62	0.37	0.67	0.22	0.42
	0.4	56.71	184.77	0.15	0.34	0.21	0.53	0.28	0.56	0.21	0.33
LLift (SVD-LLM)	0.8	7.84	10.11	0.30	0.70	0.39	0.66	0.54	0.76	0.25	0.51
	0.6	9.86	15.38	0.23	0.62	0.30	0.61	0.44	0.69	0.23	0.45
	0.4	16.88	35.33	0.17	0.46	0.25	0.58	0.34	0.61	0.23	0.38
LLift (ZS-SVD)	0.8	7.27	10.02	0.29	0.71	0.37	0.66	0.54	0.76	0.26	0.51
	0.6	8.58	14.34	0.25	0.65	0.31	0.62	0.45	0.70	0.23	0.46
	0.4	16.38	42.18	0.16	0.43	0.21	0.55	0.33	0.60	0.22	0.36

Relative to the default SVD-LLM basis, the ZS-SVD basis changes accuracy by $-0.11/+1.29/-1.75$ percentage points under ZS-SVD deployment at $r = 0.8/0.6/0.4$, and by $-0.60/-0.82/-2.32$ points under SVD-LLM deployment. Matching the preparation and deployment operators therefore does not uniformly improve transfer.

Deployment recalibration. We re-estimate activation statistics from the LLift-prepared model, apply SVD-LLM to its weights using the resulting transforms, and truncate the shared decompositions at $r \in \{0.4, 0.5, 0.6, 0.7, 0.8\}$. Table 10 reports changes relative to compressing the same weights with original-model statistics.

Table 10: Deployment recalibration of LLaMA-7B LLift under SVD-LLM. Entries are paired differences (prepared-model statistics minus original-model statistics) for the same weights. Accuracy differences use the 0–1 scale; PPL differences are absolute.

Metric	$r = 0.4$	$r = 0.5$	$r = 0.6$	$r = 0.7$	$r = 0.8$
Δ WT2 PPL	$+1.17$	$+0.83$	$+0.61$	$+0.44$	$+0.36$
Δ C4 PPL	-0.06	$+0.31$	$+0.30$	$+0.26$	$+0.19$
Δ Avg. accuracy	-0.0021	$+0.0073$	$+0.0035$	$+0.0001$	$+0.0008$

Recalibration changes mean accuracy by -0.21 to $+0.73$ percentage points. WT2 perplexity increases by 0.36–1.17 at every budget; C4 perplexity changes by -0.06 to $+0.31$ and increases at four of the five budgets. We retain original-model statistics in the main comparisons because recalibration provides no consistent perplexity benefit.

B.4 Retained Spectral Energy

We apply SVD to each $W_\ell T_\ell$ using the same frozen original-model `mix75` transforms and select prefixes according to Eq. 4. We measure retained whitened spectral energy as the fraction of squared singular values that remain after truncation. Table 11 shows that the reported retained fractions differ by at most 2×10^{-4} despite the improvements in compressed-model quality.

Table 11: Retained whitened spectral energy after truncation at each deployment budget. Both models use the same frozen `mix75` activation transforms.

Model	$r = 0.4$	$r = 0.6$	$r = 0.8$
Base	0.8818	0.9325	0.9633
LLift	0.8820	0.9327	0.9634

B.5 Gradient Alignment and Recompression Probes

We examine cross-budget transfer through gradient alignment at initialization and through the measured effect of an AdamW update followed by SVD recompression.

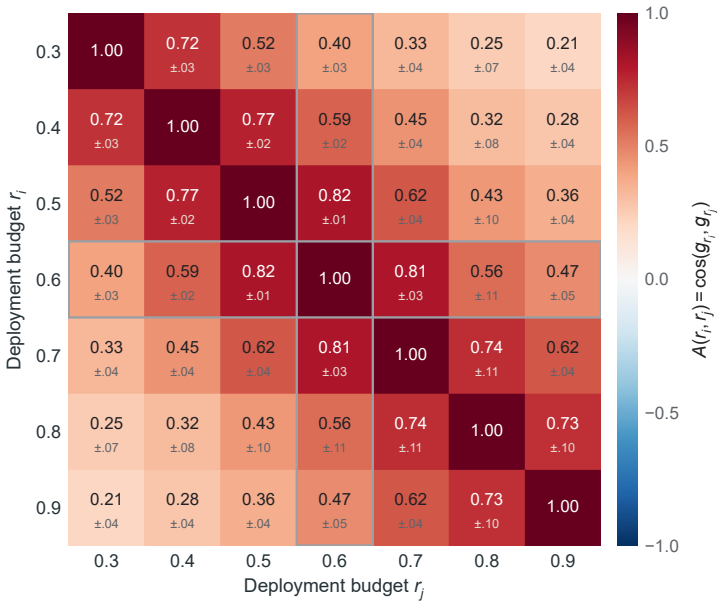


Figure 5: Cross-budget gradient alignment at initialization. Cells report mean cosine similarity and standard deviation across five disjoint WT2 batches. Figure 3(b) shows the 0.4–0.8 submatrix of means.

We use SVD-LLM projections and student-to-teacher KL from Eq. 6, with temperature one. We sample 56 of 224 target matrices across eight depths and use five disjoint WT2 batches of 32×1024 input positions. We evaluate all budgets at the original model and hold activation statistics, each budget's projector, and the remaining parameters fixed during differentiation. Let W collect the sampled target matrices and define

$$g_r = \nabla_W \widetilde{\mathcal{L}}_r(W; P_r)\Big|_{W=W_0}, \qquad A(\rho, r) = \frac{\langle g_\rho, g_r \rangle_F}{\|g_\rho\|_F \|g_r\|_F}. \tag{16}$$

Here $\widetilde{\mathcal{L}}_r$ is the fixed-projector KL surrogate, and inner products sum over the sampled matrices.

For an orthonormal basis Q_ρ of the diagnostic projector $P_\rho = Q_\rho Q_\rho^\top$ and upstream gradient G_ρ, the corresponding factor coordinates satisfy $\nabla_B \mathcal{L}_\rho = Q_\rho^\top G_\rho$. An ordinary gradient step confined to the sampled factor coordinates therefore induces the model-weight update $\Delta W = Q_\rho \Delta B = -\eta Q_\rho Q_\rho^\top G_\rho = -\eta g_\rho$. A first-order expansion of $\widetilde{\mathcal{L}}_r$ along this update gives Eq. 11 for sufficiently small η. All off-diagonal means in Figure 5 are positive, identifying a shared local descent component at initialization. Standard deviations measure variation across probe batches at the same model checkpoint.

Recompression probe and controls. For the update probe in Figure 3(c), let $\Delta\Theta$ denote the first nonzero AdamW displacement from the default $\rho = 0.6$ run. We measure $J_r(\Theta) = \mathrm{KL}(p_{\mathrm{S},r}(\Theta) \| p_{\mathrm{T}})$ on 64 held-out WT2 validation sequences of length 1024, with no overlap with training batches, at temperature one. Here $p_{\mathrm{S},r}(\Theta)$ comes from compressing Θ at budget r and p_{T} comes from the frozen original teacher. For each scale s, we recompute the weighted SVDs of the updated target matrices using the frozen original-model `mix75` deployment transforms and report

$$\Delta J_r(s) = J_r(\Theta_0 + s\Delta\Theta) - J_r(\Theta_0).$$

Table 12 reports the full update at $s \in \{0.25, 0.5, 1\}$ and the target-only and norm-matched random controls at $s = 1$; negative values indicate improvement. To isolate the target-matrix contribution, we write $\Delta\Theta = (\Delta W, \Delta\Phi)$ and repeat the probe with $(\Delta W, 0)$, keeping the remaining parameters at Φ_0.

Table 12: Change in held-out student-to-teacher KL after scaling the first nonzero AdamW update and recomputing the deployment SVD factors. Entries report $\Delta J_r(s)$ in 10^{-3} nats/token; negative values indicate improvement over compressing the original model. Signed zeros reflect rounding.

Update	s	$r = 0.4$	$r = 0.5$	$r = 0.6$	$r = 0.7$	$r = 0.8$
Full $(\Delta W, \Delta\Phi)$	0.25	-570	-508	-283	-120	-53
	0.5	-932	-793	-412	-177	-79
	1	-1244	-1007	-504	-218	-93
Target only $(\Delta W, 0)$	1	-1223	-994	-497	-213	-91
Norm-matched random	1	$+0.8$	$+0.4$	$+0.2$	-0.0	$+0.0$

At $s = 1$, the target-only changes match the full-update changes within 2.3%, while the random control stays near zero. For this first nonzero update at $s \in \{0.25, 0.5, 1\}$, we also compare KL changes under fixed projectors with those after recomputing the deployment SVD factors. The changes have the same sign; the largest reported relative difference is about 37%, at $s = 0.25$ and $r = 0.4$.

B.6 Budget-Specific Fine-Tuning

We compress the original LLaMA-7B model with SVD-LLM using WT2 calibration, then fine-tune each compressed model independently on Alpaca. We first train LoRA adapters on the `*_u_proj` factors and merge them, then train and merge adapters on the `*_v_proj` factors. We report the selected configuration from the LoRA hyperparameter search, using seed 42, approximately three epochs per stage, and a learning rate of about 10^{-4}. We evaluate mean plain accuracy over the same seven tasks.

Table 13 compares the resulting models with the single LLift model prepared at $\rho = 0.6$. At $r = 0.8/0.6/0.4$, LLift's accuracy is $0.15/0.22/1.13$ percentage points lower, while its WT2 perplexity is lower at all three budgets. LLift also has lower C4 perplexity at $r = 0.8/0.6$, while budget-specific fine-tuning has lower C4 perplexity at $r = 0.4$.

B.7 Preparation and Baseline Protocols

Target matrices and preparation. We factorize the attention projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and MLP projections (`gate_proj`, `up_proj`, `down_proj`) in every Transformer block. LLaMA-7B has 224 such matrices across 32 blocks. We group the token embed-

Table 13: Budget-specific fine-tuning and LLift on LLaMA-7B. SVD-LLM + FT uses WT2 calibration followed by two-stage LoRA fine-tuning on Alpaca at each budget; Before FT reports the accuracy of its compressed initialization. LLift uses one model prepared at $\rho = 0.6$ and frozen `mix75` deployment transforms for all budgets. WT2 and C4 report perplexity; Avg. is seven-task mean plain accuracy.

r	Before FT Avg.	SVD-LLM + FT			LLift		
		WT2 ↓	C4 ↓	Avg. ↑	WT2 ↓	C4 ↓	Avg. ↑
0.8	0.38	7.62	12.01	0.50	7.49	11.15	0.50
0.6	0.35	9.54	15.78	0.46	8.40	14.77	0.46
0.4	0.31	15.19	26.79	0.39	13.48	32.24	0.38

dings, output head, and normalization parameters into Φ and train them without factorization. For the LLaMA-7B reference, we use seed 42, teacher KL with temperature one, and the WT2 training split. We run AdamW for four epochs (176 optimizer steps) with learning rate 10^{-4}, warmup fraction 0.05, weight decay 0.01, gradient clipping at norm 1, sequence length 1024, and global batch size 64. We construct the preparation bases from original-model activations on WT2 and freeze both the teacher and these bases.

Deployment calibration and scoring. We estimate activation transforms from the original model on 256 sequences of length 2048. This calibration set contains 75% WT2 and 25% Alpaca (`mix75`); we construct it separately from preparation calibration. We freeze these transforms across models and budgets, decompose each $W_\ell T_\ell$ once, and select prefixes for each deployment budget. We evaluate WT2 perplexity on its held-out test split and C4 perplexity separately. We average unrounded plain accuracies equally over the seven tasks in Sec. 5. The paired residual-restoration control keeps the adapted non-target parameters and deployment transforms identical.

Compression and elastic baselines. We use the SVD baselines documented by Abbasi et al. (2026); Hu et al. (2026). For cross-model experiments, SVD-LLM uses 256 WT2 sequences of length 2048, and ZS-SVD uses 256 WT2 samples.

For the elastic comparison, ACIP uses C4 and FlexRank uses FineWeb-Edu for preparation, while LLift uses WT2. ACIP omits the rank-32 correction Δ; ACIP ($+\Delta$) retains it. The reported budget for ACIP ($+\Delta$) excludes this correction's 79.95M parameters.

Spectral-regularization control. For the spectral control in Table 3, we train the dense LLaMA-7B student with student-to-teacher KL at temperature one, without CE or a low-rank forward pass. We match LLift's AdamW recipe and 11.5M-token training budget described above. For each of the 224 attention and MLP target matrices $W_\ell \in \mathbb{R}^{m_\ell \times n_\ell}$, we penalize the normalized truncated Schatten-1 tail

$$\mathcal{R}_\ell(W_\ell) = \frac{\sum_{i>k_\ell} \sigma_i(W_\ell T_\ell)}{\|W_\ell T_\ell\|_F}, \qquad k_\ell = \left\lfloor \frac{0.6\, m_\ell n_\ell}{m_\ell + n_\ell} \right\rfloor. \tag{17}$$

We compute and freeze the whitening factors T_ℓ from the original model on 256 WT2 sequences of length 1024. We compute spectra from FP32 Gram eigenvalues floored at 10^{-6}. At each optimizer step, we apply a separate Muon update for the spectral term with coefficient 10^{-3}, learning rate 10^{-5}, momentum 0.95, five Newton–Schulz iterations, and RMS scale 0.2. We compress the trained weights with SVD-LLM using frozen original-model `mix75` transforms and the shared evaluation protocol described above.

Low-Rank Prehab reimplementation. We implement the Prehab objective as CE plus $\lambda \sum_\ell \|W_\ell T_\ell\|_*^2 / \|W_\ell T_\ell\|_F^2$, using frozen original-model whitening factors, $\lambda = 10^{-3}$, and learning rate 10^{-5}. After approximately 0.79M training token positions, we apply SVD-LLM with the frozen `mix75` transform and obtain $0.33/0.39/0.30$ mean accuracy at $r = 0.8/0.6/0.4$, compared with $0.44/0.40/0.31$ for the original model. This run uses less training exposure than default LLift, so we report it as an implementation diagnostic.

C Preparation Cost and Deployment Accounting

For deployment ratios $\mathcal{R}$, we account for preparation, calibration, decomposition, and per-budget factor materialization:

$$C_{\mathrm{LLift}}(\mathcal{R}) = C_{\mathrm{prep}} + C_{\mathrm{cal}} + C_{\mathrm{SVD}} + \sum_{r \in \mathcal{R}} C_{\mathrm{materialize}}(r). \tag{18}$$

With one prepared model and fixed activation transforms, we compute one weighted SVD per target matrix and materialize factors at each budget.

Table 14: Measured LLift preparation stages on A100-80GB GPUs. GPU-hours equal elapsed seconds times device count divided by 3600. The two 7B stages total 1.65 GPU-hours; the 13B row covers optimization only.

Model	Measured stage	Elapsed (s)	GPUs	GPU-h
LLaMA-7B	Optimization	1405.4	4	1.56
LLaMA-7B	Basis construction	313	1	0.09
LLaMA-13B	Optimization (CPU offload)	4472	4	4.97

Measured preparation stages. In Table 14, LLaMA-7B optimization and basis construction cost $(1405.4 \times 4 + 313)/3600 \approx 1.65$ A100 GPU-hours once for all budgets. This total excludes model export, deployment calibration and SVD, and evaluation. For LLaMA-13B, we report only optimization with CPU offload; we did not time basis construction separately.

Budget-specific fine-tuning. The two-stage LoRA procedure in Appendix B.6 takes approximately 3.6 GPU-hours per deployment budget. Summing the fine-tuning costs gives 10.8 GPU-hours for $r \in \{0.8, 0.6, 0.4\}$. These totals describe budget-specific fine-tuning; compression and deployment are accounted for separately. LLift instead incurs its preparation cost once and shares the prepared model across budgets.

Deployment and reference timings. The A100 records in Table 4 give 891.8 seconds (0.25 GPU-hours) for one-shot ZS-SVD and 7587.2 seconds (2.11 GPU-hours) for ZS-SVD $5\times$ at one budget. Both LLift entries add one-time preparation to one-budget compression: $1.65 + 0.25 = 1.90$ GPU-hours.

The ACIP reproduction reports approximately 0.87 GPU-hours without specifying hardware or stages; FlexRank reports approximately 26 GPU-hours. We account for calibration, decomposition, correction or search, and factor materialization as separate stages. We sum occupied device time, counting shared work once and excluding quality evaluation.