STRIM: STRUCTURE-AWARE LOW-RANK INFERENCE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank model compression has become an effective strategy for reducing the size of LLMs while maintaining performance. This paper introduces STRIM (STRucture-aware Low-rank Inference Model), a method for the low-rank compression of large language models. We start from a guiding block-level reconstruction view and derive a greedy, student-input-aware compression procedure. Compared with matrix-wise low-rank compression methods and recent student-input-aware approaches, STRIM explicitly accounts for compression-relevant architectural operators such as residual connections and normalization layers, which helps mitigate cascading compression errors. Experimental results show that STRIM consistently improves classification accuracy across model families (e.g., Mistral, Gemma, Qwen) and compression ratios, with gains exceeding five percentage points in several settings.

1 INTRODUCTION

The size of Large Language Models (LLMs) has grown rapidly in recent years, requiring substantial computational and memory resources for deployment. To mitigate these costs, LLM compression has become an effective strategy for reducing model size while maintaining performance. In this paper, we focus on low-rank compression, a technique that approximates an original weight matrix with two smaller low-rank matrices to save both memory and computation without requiring special hardware.

The most straightforward low-rank compression method is to apply truncated SVD to each weight matrix. While this technique has shown promise on earlier models (Shim et al., 2017; Sindhwani et al., 2015; Yu et al., 2017), it performs poorly on modern neural architectures, where weight matrices are often not inherently low-rank (Chen et al., 2021; Hsu et al., 2022). To address this limitation, several recent methods have proposed incorporating different importance weights into SVD. For instance, FWSVD (Hsu et al., 2022) uses the fisher information matrix to guide the decomposition, while ASVD (Yuan et al., 2023) leverages the norms of the rows of the empirical covariance matrix. Both DRONE (Chen et al., 2021) and SVD-LLM (Wang et al., 2025b) utilize the full empirical covariance matrix to determine importance. These methods guide the model compression to be more accurate in the denser subspace of the inputs and improve the performance.

However, despite improving over vanilla SVD, these approaches still treat each weight matrix in isolation. Other recent approaches improve upon purely matrix-wise compression by matching local activations, but such matching is often used mainly as a post-compression refinement stage. In this paper, we instead use local reconstruction to motivate a greedy, student-input-aware compression procedure. More specifically, we focus on the mismatch between the teacher hidden states from the full model and the student hidden states actually seen after earlier layers have already been compressed. We further argue that compression-relevant architectural operators should be incorporated explicitly: residual connections and normalization layers directly affect the regression targets and error propagation around compressed linear layers, and accounting for them substantially improves the compression objective, especially under more aggressive compression. Consequently, in this paper, "structure-aware" refers specifically to these operators rather than to every Transformer design choice such as attention variants including group-query or sliding-window attention. This perspective is related to local feature distillation methods such as LilLama, which also recognize the importance of student hidden states. Developed concurrently with the initial version of our work, AA-SVD also studies shifted compressed inputs. However, its recommended refined pipeline uses an original-input,

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.

SVD-LLM-style closed-form initializer and introduces student inputs only during subsequent block refinement. STRIM instead retains student inputs in its closed-form initialization and combines them with residual- and normalization-dependent targets and original-weight-anchored regularization.

Our contributions can be summarized as follows:

- We identify that low-rank compression should account for the mismatch between teacher hidden states from the full model and the student hidden states actually seen after earlier compression, and that residual connections and normalization layers further shape this mismatch.
- We propose a structure-aware low-rank compression formulation motivated by a block-level reconstruction objective. The resulting method is a greedy, student-input-aware compression procedure with explicit residual and normalization corrections and efficient regularized closed-form updates, which provide a strong initialization for the subsequent gradient-based optimization (OPT) stage.
- Extensive experiments across multiple LLMs demonstrate that our method consistently improves average accuracy across six classification benchmarks under matched parameter-size constraints, with gains exceeding five percentage points in several settings. We further evaluate STRIM under post-compression optimization and instruction finetuning settings, and show that our method remains compatible with adaptive rank selection and quantization.

2 Related Work

Low-rank Compression. Low-rank compression approximates a weight matrix with two smaller factors, most commonly via Singular Value Decomposition (SVD). Recent data-aware variants use calibration activations to guide compression, including FWSVD (Hsu et al., 2022), ASVD (Yuan et al., 2023), DRONE (Chen et al., 2021), SVD-LLM (Wang et al., 2025b), BasisSharing (Wang et al., 2025a), and PIFA (Zhao et al., 2025). Recent adjacent work also explores feature-guided rank allocation (Ji et al., 2024; Qinsi et al., 2025; Ding et al., 2025) and layer-wise suppression of accumulated and local errors (Hu et al., 2026). A key limitation of the matrix-wise line is that it compresses each weight matrix using activations from the full model and therefore does not explicitly optimize for the mismatch between teacher activations and the student activations seen after earlier layers have been compressed. Our formulation instead uses compressed inputs $\bar{X}^l$ and derives greedy updates informed by residual and RMSNorm operators.

ESPACE (Sakr & Khailany, 2024) uses a calibrated static orthonormal projection of original-model activations to induce a low-rank weight factorization through matrix-multiplication associativity. Unlike STRIM, it does not account for shifted student inputs or construct targets based on residual and normalization operators. EoRA (Liu et al., 2024) instead adds a low-rank correction C to an already compressed weight by solving $\min_{\mathrm{rank}(C)\leq r} \|\Delta W X - CX\|_F^2$. This objective is a special case of Theorem 4.1, obtained by treating C as the unknown, setting $Y = \Delta W X$, and using $\eta = 0$; STRIM instead directly constructs the compressed replacement weight from student inputs and structure-adjusted targets.

Local feature matching methods are closely related to our reconstruction-based perspective. LilLama (Sy et al., 2025) uses SVD initialization followed by local gradient-based activation matching with joint teacher/student losses. Developed concurrently with the initial version of our work, AA-SVD (Sinha & Fleuret, 2026) studies shifted compressed inputs and subsequently applies student-input-aware block refinement. However, its recommended refined pipeline uses an original-input, SVD-LLM-style closed-form initializer because its alternative student-input-aware initializer does not improve performance after refinement. That alternative initializer corresponds closely to STRIM's generic linear case in Equation (4). STRIM instead retains student inputs in its closed-form initialization and additionally constructs residual- and normalization-dependent targets before refinement. Thus, the methods use related reduced-rank regression machinery but differ in how the initialization targets are constructed.

Other Compression Methods. Beyond dense low-rank compression, common alternatives include pruning (Ma et al., 2023; Zhong et al., 2024) and quantization (Dettmers et al., 2023; Lin et al., 2024). Related hybrid methods combine low-rank approximation with sparsity or low precision, including LoSparse (Li et al., 2023), LoRAP (Li et al., 2024), low-rank low-precision decomposition

(Saha et al., 2024), and SLiM (Mozaffari et al., 2025); these are outside the scope of our main dense low-rank comparison. Our method is complementary to quantization, as shown in Table 8.

3 BACKGROUND

Modern dense decoder-only LLMs are stacks of transformer blocks with residual streams. Each block contains attention and feed-forward sublayers, and recent architectures widely use RMSNorm or related normalization around these sublayers (Zhang & Sennrich, 2019). In this paper, residual pathways and normalization are the architectural operators that most directly affect local reconstruction and error propagation for compressed linear layers.

The goal of low-rank LLM compression is to approximate each weight matrix with a low-rank counterpart. When considering a weight matrix W_h^l of layer l in isolation, its optimal low-rank approximation under the Frobenius norm is given by truncated Singular Value Decomposition (SVD):

$$\bar{W}_h^l = \mathrm{SVD}_r(W_h^l), \tag{1}$$

where r is the target rank. This reduces the parameter count and FLOPs from $O(mn)$ to $O(r(m+n))$. However, plain SVD ignores the input distribution X_h^l seen by each weight matrix. Motivated by this, several low-rank compression methods use a calibration set to estimate the empirical distribution of X_h^l and perform more data-aware compression (Hsu et al., 2022; Yuan et al., 2023; Chen et al., 2021; Wang et al., 2025b). Our method starts from the same low-rank setting but moves from matrix-wise compression toward a greedy, student-input-aware procedure motivated by a block-level objective.

4 PROPOSED METHOD

A Transformer block refers to one complete decoder block, consisting of a self-attention component and a feed-forward (FFN) component. We use *Transformer sub-block* to refer to either component, including its linear projections and the associated residual and normalization operators relevant to compression. Instead of compressing each layer in isolation, we use a block-level objective over the entire Transformer sub-block as a guiding formulation. For each sub-block f^l, we consider the block MSE objective

$$\min_{\bar{f}^l \in \Pi} \|\bar{f}^l(\bar{X}^l) - f^l(X^l)\|_F^2, \tag{2}$$

where X^l and $\bar{X}^l$ are the inputs to the original and compressed sub-blocks, respectively. The set Π contains all compressed sub-blocks $\bar{f}^l$ whose weight matrices satisfy the prescribed rank constraints.

For general f^l, there is no closed-form joint solution for (2). We therefore adopt a greedy strategy that sequentially minimizes the output error of each weight matrix as an approximation to this block-level objective. Specifically, we define a sequence of sublayers h, each parameterized by a single currently uncompressed weight matrix W_h together with a selection of previously compressed matrices. For each sublayer, we solve for its compressed counterpart $\bar{h}$ by optimizing

$$\min_{\bar{W}_{\bar{h}} \in \Pi_r} \|\bar{h}(\bar{X}^l) - h(X^l)\|_F^2, \tag{3}$$

where Π_r is the low-rank constraint. For most weight matrices W_h, the sublayer can be written as a linear mapping $h(X^l) = W_h g(X^l)$, where $g(X^l)$ is the sublayer input. Compressing W_h then reduces to the least-squares problem

$$\min_{\bar{W}_{\bar{h}} \in \Pi_r} \|\bar{W}_{\bar{h}} \bar{g}(\bar{X}^l) - W_h g(X^l)\|_F^2. \tag{4}$$

Two common structures in LLMs, however, require special treatment: residual connections and normalization.

Residual Connection. Residual connections (He et al., 2016) are standard in modern LLMs, including Mistral (Jiang et al., 2024), Gemma (GemmaTeam, 2025), and Qwen3 (Yang et al., 2025). Let h be a sublayer of f^l of the form

$$h(X^l) = r_{\text{res}}(X^l) + W_h g(X^l),$$

where r_{res} is the residual connection. Substituting this into (3) gives

$$\min_{\bar{W}_{\bar{h}} \in \Pi_r} \|\bar{r}_{\text{res}}(\bar{X}^l) + \bar{W}_{\bar{h}} \bar{g}(\bar{X}^l) - r_{\text{res}}(X^l) - W_h g(X^l)\|_F^2. \tag{5}$$

Compared to (4), this objective introduces an extra term $\bar{r}_{\text{res}}(\bar{X}^l) - r_{\text{res}}(X^l)$, which we call the residual connection error. This term becomes significant as the compression error accumulates across the transformer blocks. This structure appears in pre-layer-normalization transformers for the output projection matrix W_{output} of the attention layer and the final projection matrix W_{down} of the FFN layer.

Normalization. Root mean square normalization (RMSNorm) is widely used in recent transformer architectures (GemmaTeam, 2025; Yang et al., 2025; Bi et al., 2024). Let h be a sublayer of f^l of the form

$$h(X^l) = \text{RMSNorm}(W_h g(X^l)) \equiv W_h g(X^l)/\text{RMS}(W_h g(X^l)),$$

where RMS$(\cdot)$ calculates the root mean square for each sample.

Substituting this into (3) gives

$$\min_{\bar{W}_{\bar{h}} \in \Pi_r} \|\bar{W}_{\bar{h}} \bar{g}(\bar{X}^l)/\text{RMS}(\bar{W}_{\bar{h}} \bar{g}(\bar{X}^l)) - W_h g(X^l)/\text{RMS}(W_h g(X^l))\|_F^2. \tag{6}$$

Compared to (4), this formulation includes additional per-sample scalings. These scalings are crucial as the RMSNorm layer plays an important role in the transformer architecture. This structure appears in post-layer-normalization transformers such as Gemma-3 (GemmaTeam, 2025) for the final projection matrix W_{output} of the attention layer and W_{down} of the FFN layer, and also for the query and key projection matrices in transformers that adopt Query-Key-Normalization (QK-norm), such as Gemma-3 (GemmaTeam, 2025) and Qwen-3 (Yang et al., 2025).

Regularization. We further observe that regularization is important to avoid overfitting to the sampled calibration set. We therefore consider the reduced-rank regression problem

$$\min_{\bar{W}_{\bar{h}}; \text{rank}(\bar{W}_{\bar{h}}) \le r} \|\bar{W}_{\bar{h}} X - Y\|_F^2 + \eta \|\bar{W}_{\bar{h}} - W_h\|_F^2, \tag{7}$$

where $\eta \ge 0$ is a hyperparameter that controls the regularization strength. We find that this performs better than the standard ℓ_2 regularization

$$\min_{\bar{W}_{\bar{h}}; \text{rank}(\bar{W}_{\bar{h}}) \le r} \|\bar{W}_{\bar{h}} X - Y\|_F^2 + \eta \|\bar{W}_{\bar{h}}\|_F^2, \tag{8}$$

which unnecessarily suppresses components of $\bar{W}_{\bar{h}}$ that are also present in W_h. To normalize the regularization strength by the input energy, we use $\eta = \eta_0 \,\text{tr}(XX^\top)/d = \eta_0 \|X\|_F^2/d$, where d is the input dimension.

Offline calibration workflow. STRIM is applied during an offline calibration phase using the fixed calibration set described in Section 5.1. We process weight matrices greedily in forward-computation order. For each matrix, we construct X and Y from the full teacher and the current partially compressed student over all calibration samples, so they reflect the effects of matrices compressed earlier in the sequence, and solve the resulting regression once to initialize the compressed weight. The closed-form problem is not solved separately for individual samples or inference inputs, nor is it repeatedly solved during the optional OPT stage.

Efficient low-rank closed form solutions. We show that (4), (5), and (6) can all be written in the form of (7). For (4), we have $X = \bar{g}(\bar{X}^l)$ and $Y = W_h g(X^l)$. Similarly, (5) can be written as (7) with $X = \bar{g}(\bar{X}^l)$ and $Y = W_h g(X^l) + r_{\text{res}}(X^l) - \bar{r}_{\text{res}}(\bar{X}^l)$.

For the RMSNorm case (6), we approximate the sample-wise root mean squares of the compressed and full-model outputs as

$$\text{RMS}(\bar{W}_{\bar{h}} \bar{g}(\bar{X}^l)) \approx \text{RMS}(W_h g(X^l)),$$

and incorporate this per-sample scaling by setting $X = \bar{g}(\bar{X}^l)/\boldsymbol{c}$ and $Y = W_h g(X^l)/\boldsymbol{c}$, where $\boldsymbol{c} = \text{RMS}(W_h g(X^l))$ is a per-sample scaling vector. The equations suppress RMSNorm's fixed learned scale and stabilizer, which are retained in implementation; the normalization case takes precedence when a projection is normalized before residual addition.

The resulting closed form is given by the following theorem; the proof is in the appendix.

Theorem 4.1. *An optimal solution to* (7) *is*

$$\bar{W}_{\bar{h}} = \mathrm{SVD}_r((\hat{Y}\hat{X}^\dagger)\hat{X})\hat{X}^\dagger, \tag{9}$$

where $\hat{X} = \mathrm{concat}([X, \eta^{1/2} I])$ *concatenates* X *with an identity matrix and* $\hat{Y} = \mathrm{concat}([Y, \eta^{1/2} W_h])$ *concatenates* Y *with the original weight* W_h *along the second dimension, and* $\hat{X}^\dagger$ *is the pseudo-inverse of a matrix* $\hat{X}$.

This result extends the reduced-rank linear regression solution of Izenman (1975), which gives the closed form of $\min_{\bar{W}_{\bar{h}};\mathrm{rank}(\bar{W}_{\bar{h}})\le r} \|\bar{W}_{\bar{h}} X - Y\|_F^2$ without the regularization term.

This closed-form solution differs from those in DRONE and SVD-LLM because our X and Y depend on both X^l and $\bar{X}^l$, whereas DRONE and SVD-LLM only consider X^l.

Complexity. For efficient computation of (9), let $S = (\hat{X}\hat{X}^\top)^{-1/2}$ denote the inverse square root of the empirical covariance matrix $\hat{X}\hat{X}^\top$ for $\eta > 0$; for $\eta = 0$, we use its pseudoinverse analogue. Then

$$\mathrm{SVD}_r(\hat{Y}\hat{X}^\dagger\hat{X})\hat{X}^\dagger = \mathrm{SVD}_r(\hat{Y}\hat{X}^\top(\hat{X}^\dagger)^\top)\hat{X}^\dagger = \mathrm{SVD}_r(\hat{Y}\hat{X}^\top S)S.$$

The computation time of S is $O(nd^2 + d^3)$. Assuming without loss of generality that the output dimension of Y is a constant multiple of d, the computation time of $\hat{Y}\hat{X}^\top S$ is also $O(nd^2 + d^3)$, while computing its SVD takes $O(d^3)$. Consequently, the total time complexity for STRIM is $O(nd^2 + d^3 + O_{\text{forward}})$, where O_{forward} is the complexity to compute X and Y. For memory, note that $\hat{X}\hat{X}^\top$ and $\hat{Y}\hat{X}^\top$ can be computed streamingly without storing the full $\hat{X}$ and $\hat{Y}$ in memory, yielding complexity $O(d^2)$. These complexities are identical to those of SVD-LLM (Wang et al., 2025b). STRIM has the same asymptotic closed-form cost as SVD-LLM, while collecting partial-student activations adds offline computation within O_{forward} but no inference-time overhead; OPT adds further offline cost.

The greedy order follows the model's forward computation graph rather than a learned or tuned heuristic. Blocks are processed from shallow to deep; within each LLaMA-like block, the order is

$$\{W_q, W_k, W_v\} \to W_o \to \{W_{\text{gate}}, W_{\text{up}}\} \to W_{\text{down}}.$$

Matrices within braces are parallel and receive the same inputs, so their internal order does not affect the collected regression inputs. Other architectures are processed analogously according to their forward graphs. This is the same ordering used by AA-SVD.

Algorithm 1 summarizes our proposed method.

Algorithm 1 STRIM

for each W_h in forward-computation order **do**
  Obtain the input $g(X^l)$
  Obtain the target rank r for the compressed weight $\bar{W}_{\bar{h}}$
  if the output of W_h is normalized by RMSNorm **then**
    # e.g. W_{query}, W_{key} with QK-norm, and W_{output}, W_{down} for post-layer norm LLMs
    Set $\boldsymbol{c} = \mathrm{RMS}(W_h g(X^l))$ for each sample
    Set $X = \bar{g}(\bar{X}^l)/\boldsymbol{c}$ and $Y = W_h g(X^l)/\boldsymbol{c}$
  else if the output of W_h is added with a residual $r_{\text{res}}(X^l)$ **then**
    # e.g. W_{output}, W_{down} for pre-layer norm LLMs
    Set $X = \bar{g}(\bar{X}^l)$ and $Y = W_h g(X^l) + r_{\text{res}}(X^l) - \bar{r}_{\text{res}}(\bar{X}^l)$
  else
    Set $X = \bar{g}(\bar{X}^l)$ and $Y = W_h g(X^l)$
  end if
  Compute $\hat{X} = \mathrm{concat}([X, \eta^{1/2} I])$ and $\hat{Y} = \mathrm{concat}([Y, \eta^{1/2} W_h])$
  Compute $S = (\hat{X}\hat{X}^\top)^{-1/2}$
  Obtain the compressed weight $\bar{W}_{\bar{h}} = \mathrm{SVD}_r(\hat{Y}\hat{X}^\top S)S$
end for

5 EXPERIMENTS

5.1 SETUP

Comparing Methods. We compare our proposed method primarily against prior calibration-time low-rank compression baselines: FWSVD (Hsu et al., 2022), ASVD (Yuan et al., 2023), SVD-LLM (Wang et al., 2025b), AdaSVD (Li et al., 2025), BasisSharing (Wang et al., 2025a), PIFA (Zhao et al., 2025), and AA-SVD (Sinha & Fleuret, 2026), along with model pruning baselines including LLM-Pruner (Ma et al., 2023) and BlockPruner (Zhong et al., 2024). Among these methods, AA-SVD is a particularly relevant compression-time baseline: its recommended pipeline uses an original-input initializer followed by block refinement with shifted student inputs. AA-SVD also reports stronger results than several recent SVD-based variants, including Dobi-SVD (Qinsi et al., 2025), Dip-SVD (Ding et al., 2025), and SAES-SVD (Hu et al., 2026), which further motivates treating it as a strong representative baseline from this line of work. We also note the closely related LilLama (Sy et al., 2025), which uses SVD initialization followed by post-compression local activation matching; we discuss this family separately in Related Work. We do not include SVD-LLM v2 as a baseline because its code is not publicly released, and in our attempt to follow the published rank-allocation formulation on the model families studied here, the resulting allocation produced negative ranks for some layers. Our empirical study focuses on low-rank compression for dense decoder-only LLMs, so MoE-specific compression methods are outside the scope of our main comparison.

In this paper, we focus on achieving a better low-rank approximation with a given parameter ratio constraint. However, among the low-rank compression baselines, ASVD, SVD-LLM, and AdaSVD include mechanisms to adaptively assign different parameter ratios to each matrix. To demonstrate the potential of combining an adaptive ratio approach with our method, we adopt a simple heuristic: we increase the parameter ratios for the key and value projection matrices while reducing them for the query and output projection matrices. This simple heuristic improves results across the Mistral compression ratios, and we name this variant STRIM+. We view STRIM+ as a practical extension of the base STRIM formulation rather than as a separately derived optimal allocation rule. We emphasize that the main focus of this paper is the compression objective and the resulting low-rank updates, rather than a new principled rank selection algorithm; rank allocation is therefore treated as an orthogonal design choice. We list the details on the hyperparameter in Appendix A.2. Our main comparisons are conducted under matched parameter-ratio constraints, and for STRIM versus SVD-LLM we use the same rank allocation; thus the comparison isolates the quality of the compressed weights rather than changes to the deployed low-rank architecture.

Implementations. For STRIM and STRIM+, unless otherwise specified, we conduct 10 epochs of gradient-based optimization on the block MSE objective (2) with Adam (batch_size=1, lr=1e-7), since we find that smaller batch sizes perform more consistently across settings, consistent with Marek et al. (2025). We set $\eta_0 = 1$ for the regularization parameter in the optimized setting, and $\eta_0 = 0.1$ for the corresponding no-OPT evaluations. We also report results without such optimization to assess the quality of the closed-form initialization. Unless otherwise stated, all baseline results are reproduced by us using the authors' released code. For AdaSVD, we use our own implementation following the published method. Our AA-SVD results use the authors' released code and recommended configuration. Supporting Mistral required only registering `MistralForCausalLM` with its existing LLaMA adapter, a 15-line change; we did not modify its compression objective, closed-form solver, matrix ordering, rank allocation, optimizer, or refinement procedure. We evaluate AA-SVD on the same Mistral-7B checkpoint, WikiText-2 calibration setting, parameter ratios, and evaluation protocol as the other methods. All experiments are conducted with PyTorch and Huggingface Transformers on a single NVIDIA A100-40G GPU. A representative compression run takes about 5 hours on this setup, though the exact runtime varies with model size, compression ratio, and optimization budget.

Evaluation Protocols. We evaluate six open-source pretrained LLMs: gemma-3-1b (GemmaTeam, 2025), gemma-3-4b (GemmaTeam, 2025), Qwen3-8B (Yang et al., 2025), OPT-6.7B (Zhang et al., 2022), DeepSeek-llm-7b (Bi et al., 2024), and Mistral-7B (Jiang et al., 2024). We follow the evaluation setup of recent low-rank compression methods, such as ASVD (Yuan et al., 2023), SVD-LLM (Wang et al., 2025b), and AdaSVD (Li et al., 2025). Specifically, for the calibration set, we randomly select 256 samples of length 2048 from the WikiText-2 (Merity et al., 2016) to conduct model compression. For evaluation tasks, we use the LM-Evaluation-Harness Framework (Gao et al., 2021) which

Table 1: Experimental results on Mistral-7B across different parameter ratios.

Ratio	Method	Rel. RMSE ↓	PPL ↓	Classification Accuracy ↑							Generation ↑	
		WikiText-2		Openb.	ArcE.	WinoG.	HellaS.	PIQA	MathQA	Avg.	TruthQA	GSM8k
1.0	Original	0.00	5.32	34.60	80.26	74.90	61.48	80.63	36.08	61.33	40.76	38.29
0.8	FWSVD	0.79	25.49	24.20	64.69	65.98	38.72	70.24	22.08	47.65	25.95	0.30
	ASVD	0.93	21.38	21.40	57.03	59.12	36.22	67.46	24.76	44.33	24.11	1.14
	SVD-LLM	0.46	7.62	27.20	68.94	67.72	41.42	69.37	28.74	50.57	34.64	4.32
	AdaSVD	0.43	7.62	27.20	68.48	67.72	41.61	69.48	28.91	50.57	33.17	3.79
	BasisSharing	0.70	7.61	28.20	70.50	67.32	43.20	69.70	29.08	51.33	**50.31**	2.96
	AA-SVD	**0.25**	**6.60**	27.00	73.65	63.46	49.23	72.31	31.29	52.82	39.29	2.65
	STRIM	**0.25**	6.73	31.80	74.92	69.46	51.87	75.35	31.93	55.89	41.74	12.43
	STRIM+	0.28	6.68	**32.20**	**75.93**	**70.40**	**53.24**	**75.95**	**32.33**	**56.68**	45.90	**13.04**
0.6	FWSVD	0.99	238.09	14.40	34.01	53.91	27.83	54.68	19.50	34.06	13.95	0.23
	ASVD	1.20	2900.55	13.40	27.15	49.57	26.06	51.31	21.68	31.53	13.34	0.08
	SVD-LLM	0.69	16.14	19.40	42.63	55.41	30.45	58.00	22.38	38.05	3.06	1.82
	AdaSVD	0.66	15.64	19.40	42.85	59.67	31.81	59.79	22.38	39.32	7.71	1.14
	BasisSharing	0.94	15.87	17.60	45.20	56.99	31.49	59.25	22.61	38.86	27.54	1.59
	AA-SVD	**0.34**	**8.80**	21.20	61.62	61.25	39.35	64.85	25.43	45.62	31.95	2.50
	STRIM	0.37	9.20	25.80	64.48	**64.33**	41.11	67.30	26.87	48.32	30.84	**2.20**
	STRIM+	0.37	9.05	**27.20**	**67.42**	64.17	**42.84**	**68.72**	**27.10**	**49.58**	**32.44**	2.12
0.4	FWSVD	1.02	1528.48	12.80	26.98	48.70	25.76	52.45	18.63	30.89	19.46	0.00
	ASVD	1.12	4581.07	14.80	25.38	48.78	25.86	51.58	19.20	30.93	**29.38**	0.00
	SVD-LLM	0.92	87.25	12.20	28.07	49.88	26.46	52.29	20.37	31.55	1.59	0.23
	AdaSVD	0.89	70.11	13.20	28.28	52.80	26.49	52.07	20.27	32.19	1.10	0.15
	BasisSharing	1.08	65.91	12.00	29.17	52.01	26.98	52.50	20.74	32.23	19.46	0.08
	AA-SVD	**0.44**	18.69	16.60	40.57	53.51	30.49	56.86	23.05	36.85	12.85	0.99
	STRIM	0.47	18.87	**19.00**	42.47	**56.20**	31.59	57.24	22.24	38.12	12.73	1.29
	STRIM+	0.47	**18.29**	18.80	**45.29**	55.80	**31.75**	**58.00**	**22.68**	**38.72**	8.08	**2.20**
0.2	FWSVD	1.05	4074.49	13.60	26.73	49.64	25.44	51.63	18.19	30.87	22.03	0.00
	ASVD	1.63	7671.85	14.20	26.35	51.22	25.55	51.36	17.76	31.07	23.99	0.00
	SVD-LLM	1.01	1062.89	13.20	27.10	49.33	25.80	52.39	20.60	31.40	0.00	0.00
	AdaSVD	0.99	708.04	13.20	26.94	47.99	25.91	51.52	21.07	31.11	0.00	0.00
	BasisSharing	1.21	651.25	12.20	27.48	48.54	25.71	52.56	**21.37**	31.31	4.65	**1.29**
	AA-SVD	0.60	140.36	16.40	28.41	51.22	26.23	52.01	20.57	32.47	20.56	0.76
	STRIM	0.59	105.24	**15.60**	28.96	49.64	26.12	**53.75**	20.74	32.47	22.77	0.53
	STRIM+	**0.58**	**90.02**	15.00	**29.88**	**51.78**	**26.56**	53.65	20.77	**32.94**	**25.58**	0.91

Table 2: Effect of structural correction across compression rates with and without gradient-based optimization.

Method	Param Rate 0.8	Param Rate 0.6	Param Rate 0.4	Param Rate 0.2
	WikiText-2 (Perplexity ↓) / Classification (6-Task Avg. ↑)			
STRIM (no OPT)	6.94 / 52.80	9.79 / 41.75	19.47 / 35.12	79.44 / 32.45
STRIM (no OPT, w/o structural-correction)	7.39 / 50.57	13.38 / 38.71	41.21 / 33.93	260.47 / 31.98
STRIM (OPT)	6.73 / 55.89	9.20 / 48.32	18.87 / 38.12	105.24 / 32.47
STRIM (OPT, w/o structural-correction)	6.89 / 55.84	10.26 / 48.16	28.77 / 36.56	277.27 / 32.29

includes six classification datasets (OpenbookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ArcE (Clark et al., 2018), PIQA (Bisk et al., 2020), MathQA (Amini et al., 2019)) and two generation datasets (TruthfulQA (Lin et al., 2021) and GSM8k (Cobbe et al., 2021)).

For different metrics, we use ↑ to indicate the higher the better and ↓ to indicate the lower the better. Since STRIM modifies the compressed weights rather than the served low-rank architecture, we focus on compression quality rather than end-to-end serving efficiency.

5.2 Main Results

We evaluate our method across multiple parameter ratios on Mistral-7B in Table 1. Here, "Avg." denotes the average over the six classification tasks only (OpenbookQA, ArcE, WinoGrande, Hel-

Table 3: Experimental results across different large language models at parameter ratio 0.8.

Method	Gemma-3-1B	Gemma-3-4B	OPT-6.7B	DeepSeek-7B	Qwen3-8B
	WikiText-2 (Perplexity ↓)		/	Classification (6-Task Avg. ↑)	
Original	10.58 / 51.18	7.39 / 60.34	10.90 / 51.54	6.89 / 57.16	9.79 / 61.05
SVD-LLM	51.29 / 34.97	15.40 / 43.09	12.11 / 50.00	9.20 / 49.35	14.02 / 51.16
AdaSVD	22.06 / 39.39	15.40 / 44.01	11.81 / **50.40**	9.20 / 49.14	13.59 / 51.60
STRIM+	**19.47 / 41.96**	**11.62 / 49.03**	**11.60 / 50.40**	**8.50 / 52.40**	**14.24 / 54.03**

laSwag, PIQA, and MathQA), while TruthfulQA and GSM8K are reported separately. Inspired by LilLama-style local feature matching (Sy et al., 2025), we also tested a plain-SVD initialization followed by the same OPT refinement, as well as a preliminary variant that mixes both student and teacher inputs during OPT; neither was helpful enough to include as a main baseline in this comparison.

Even with a fixed parameter ratio across all weight matrices, STRIM outperforms prior low-rank compression baselines in downstream accuracy. AA-SVD is a particularly relevant comparison: its recommended closed-form initializer uses original inputs, while its subsequent block refinement uses shifted student inputs. STRIM matches or exceeds AA-SVD in six-task average classification accuracy across all parameter ratios, and STRIM+ improves this margin further.

We emphasize that this is an end-to-end comparison rather than a component-isolated attribution. Besides the explicit treatment of residual and normalization structure, the methods also differ in regularization and optimization details. Nevertheless, our overall formulation yields a stronger downstream accuracy trade-off in this setting, even though STRIM uses 10 epochs of optimization while AA-SVD uses 25 epochs of block-level refinement in its released implementation. Compared with matrix-wise baselines such as SVD-LLM, AdaSVD, and BasisSharing, our method also substantially reduces the relative root mean squared error (Rel. RMSE), $\|\bar{f}^l(\bar{X}^l) - f^l(X^l)\|_F / \|f^l(X^l)\|_F$, consistent with the reconstruction objective. AA-SVD attains slightly lower final-layer Rel. RMSE, consistent with its longer local refinement schedule, but STRIM still yields stronger downstream accuracy. (We only report final-layer Rel. RMSE here; the layer-wise comparison is provided in Appendix A.10.)

In particular, compared to SVD-LLM, STRIM improves classification accuracy by more than five percentage points at parameter ratios between 0.4 and 0.8 and improves GSM8K exact-match accuracy by over 8 percentage points at a 0.8 compression ratio. Furthermore, performance can be enhanced by incorporating adaptive rank selection (STRIM+). This demonstrates the compatibility of our method with adaptive rank selection.

We next investigate the impact of the number of calibration samples. As shown in Figure 1, both methods benefit from more calibration samples, while STRIM maintains its advantage and already outperforms SVD-LLM with only 64 samples.

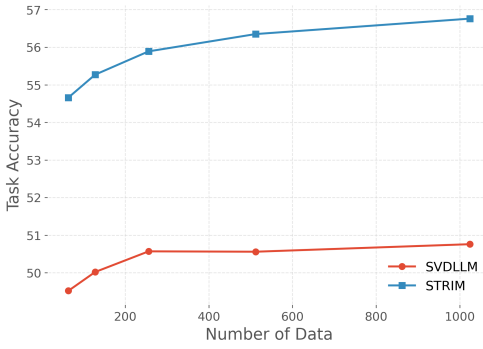

Figure 1: Comparing STRIM and SVD-LLM compressed on different numbers of calibration samples.

We then isolate one of the core ingredients of STRIM by studying structural correction across compression rates, both with and without gradient-based optimization. As shown in Table 2, structural correction consistently improves perplexity across all parameter ratios, and the gap becomes much larger as compression becomes more aggressive. The downstream accuracy gains are clearest in the no-OPT regime and remain visible after optimization, especially at lower parameter ratios. This supports the view that structural correction provides a better-aligned initialization before optimization, while the subsequent optimization stage can partially absorb the mismatch under milder compression. Because Mistral does not instantiate the output-normalization

Table 4: Optimization trajectory comparison at selected epochs. Lower local fit alone does not guarantee better downstream accuracy.

Method	Epoch 10		Epoch 100	
	Rel. RMSE ↓	6-Task Avg. ↑	Rel. RMSE ↓	6-Task Avg. ↑
STRIM	0.2500	55.89	0.2357	56.57
SVD-LLM+OPT	0.3077	54.63	0.2353	55.14

pattern treated in (6), this experiment isolates residual correction rather than combining it with RMSNorm scaling.

Appendix A.4 shows that STRIM is compatible with PIFA's pivoting factorization and that combining them yields further gains.

Table 3 extends the end-to-end comparison to multiple LLM families and sizes: STRIM+ improves six-task accuracy over SVD-LLM and matches or improves AdaSVD across all five models. Appendix A.3 provides a controlled comparison of plain STRIM and SVD-LLM+OPT under identical fixed ranks and the same OPT schedule.

5.3 Post-compression Optimization and the Role of Closed-Form Initialization

We study both post-compression optimization and the role of the proposed closed-form initialization. First, we evaluate STRIM without any gradient-based optimization on the block MSE objective. The no-OPT rows in Table 2 already show strong performance before optimization, highlighting the quality of the closed-form initialization itself. Conversely, we test the OPT baseline, which applies gradient-based optimization to SVD-LLM while omitting our closed-form initialization. As summarized in Table 4, SVD-LLM+OPT eventually reaches a nearly identical local Rel. RMSE to STRIM, and even slightly surpasses it at epoch 100 (0.2353 vs. 0.2357), yet it still remains worse on downstream accuracy (55.14 vs. 56.57). This indicates that local reconstruction error alone is not a sufficient model-selection criterion for compressed LLMs. In practice, we therefore report results under a fixed optimization budget rather than selecting checkpoints by local Rel. RMSE, and identifying a better calibration-only proxy for downstream quality is an interesting direction for future work. Instead, the structure-aware STRIM initialization appears to better preserve downstream behavior than optimizing the local objective alone. Appendix A.8 compares downstream accuracy and local Rel. RMSE across runs with different optimization budgets, showing that the downstream gap persists even as the local reconstruction gap narrows. Together, these observations show that both the proposed initialization and the subsequent optimization stage are important parts of the overall STRIM pipeline. Additional results under instruction finetuning are provided in Appendix A.5, where STRIM remains stronger than the baselines across parameter ratios.

Compatibility with other compression methods. Appendices A.7 and A.6 show that STRIM outperforms the evaluated pruning baselines in their respective settings and can be combined with AWQ quantization with little loss in accuracy.

Appendix A.9 further shows that original-weight regularization also contributes in the no-OPT setting.

6 Conclusion

We introduce STRIM, a low-rank LLM compression method that uses a block-level reconstruction view to motivate greedy closed-form updates with explicit residual and normalization correction. Across model families and compression ratios, STRIM consistently improves downstream accuracy, with gains exceeding five percentage points in several settings, while our ablations show that structural correction yields a better-aligned initialization, especially under stronger compression. The method is also compatible with adaptive rank allocation, post-compression refinement, and further compression techniques such as quantization, making it a useful component for broader low-rank compression pipelines.

AI USE STATEMENT

Generative AI tools were used primarily to improve the clarity, grammar, organization, and presentation of the manuscript, and to provide feedback on the presentation of experiments and ablations. All AI-assisted text and suggestions were reviewed and revised by the authors. The authors independently verified the mathematical derivations, citations, implementations, and reported experimental results, conducted all experiments, and made all final research decisions. We take full responsibility for the content of this work.

ETHICS STATEMENT

This work studies post-training compression of publicly available pretrained language models using public benchmarks and does not involve human subjects or the collection of personal data. Compressed models may retain harmful, biased, or misleading behaviors from their original models; therefore, safety-critical behaviors should be re-evaluated before deploying a compressed model.

REPRODUCIBILITY STATEMENT

STRIM is described in Section 4, with pseudocode provided in Algorithm 1 and the proof of the closed-form solution provided in the appendix. The experimental setup in Section 5.1 specifies the calibration data, models, baselines, evaluation metrics, hyperparameters, and compute resources. Additional experimental details and ablations are reported in the appendix. We will publicly release our source code upon acceptance.

REFERENCES

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Drone: Data-aware low-rank compression for large nlp models. *Advances in neural information processing systems*, 34:29321–29334, 2021.

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. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Xuan Ding, Rui Sun, Yunjian Zhang, Xiu Yan, Yueqi Zhou, Kaihao Huang, Suzhong Fu, Chuanlong Xie, and Yao Zhu. Dipsvd: Dual-importance protected svd for efficient llm compression. *arXiv preprint arXiv:2506.20353*, 2025.

Leo Gao, Jonathan Tow, Stella Biderman, Shawn Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jasmine Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 10:8–9, 2021.

GemmaTeam. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022.

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. In *The Fourteenth International Conference on Learning Representations*, 2026.

Alan Julian Izenman. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, 5(2):248–264, 1975.

Yixin Ji, Yang Xiang, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, Kehai Chen, and Min Zhang. Adaptive feature-based low-rank compression of large language models via bayesian optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4152–4168, 2024.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. Mistral 7b. arxiv 2023. *arXiv preprint arXiv:2310.06825*, 2024.

Guangyan Li, Yongqiang Tang, and Wensheng Zhang. Lorap: Transformer sub-layers deserve differentiated structured compression for large language models. In *International Conference on Machine Learning*, pp. 28657–28672. PMLR, 2024.

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 *International Conference on Machine Learning*, pp. 20336–20350. PMLR, 2023.

Zhiteng Li, Mingyuan Xia, Jingyuan Zhang, Zheng Hui, Linghe Kong, Yulun Zhang, and Xiaokang Yang. Adasvd: Adaptive singular value decomposition for large language models. *arXiv preprint arXiv:2502.01403*, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*, 2024.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, et al. Eora: Fine-tuning-free compensation for compressed llm with eigenspace low-rank approximation. *arXiv preprint arXiv:2410.21271*, 2024.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum. Small batch size training for language models: When vanilla sgd works, and why gradient accumulation is wasteful. *arXiv preprint arXiv:2507.07101*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. Slim: One-shot quantization and sparsity with low-rank approximation for llm weight compression. In *International Conference on Machine Learning*, pp. 45024–45049. PMLR, 2025.

Wang Qinsi, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Dobi-svd: Differentiable svd for llm compression and some new perspectives. In *The Thirteenth International Conference on Learning Representations*, 2025.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37:88981–89018, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Charbel Sakr and Brucek Khailany. Espace: Dimensionality reduction of activations for model compression. *Advances in Neural Information Processing Systems*, 37:17489–17517, 2024.

Kyuhong Shim, Minjae Lee, Iksoo Choi, Yoonho Boo, and Wonyong Sung. Svd-softmax: Fast softmax approximation on large vocabulary neural networks. *Advances in neural information processing systems*, 30, 2017.

Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. *Advances in Neural Information Processing Systems*, 28, 2015.

Atul Kumar Sinha and François Fleuret. AA-SVD: Anchored and adaptive SVD for large language model compression. *arXiv preprint arXiv:2604.02119*, 2026.

Yaya Sy, Christophe Cerisara, and Irina Illina. Efficient one-shot compression via low-rank local feature distillation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5643–5661, 2025.

Jingcun Wang, Yu-Guang Chen, Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7370–7379, 2017.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Jialin Zhao, Yingtao Zhang, and Carlo Vittorio Cannistraci. Pivoting factorization: A compact meta low-rank representation of sparsity for efficient inference in large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. BlockPruner: Fine-grained pruning for large language models. *arXiv preprint arXiv:2406.10594*, 2024.

A Appendix

A.1 Proof of Theorem 4.1

By our definition

$$\hat{X} = \text{concat}([X, \eta^{1/2} I]), \hat{Y} = \text{concat}([Y, \eta^{1/2} W_h]),$$

thus we have

$$\begin{aligned} &\|\bar{W}_{\bar{h}} X - Y\|_F^2 + \eta \|\bar{W}_{\bar{h}} - W_h\|_F^2 = \|\bar{W}_{\bar{h}} X - Y\|_F^2 + \|\eta^{1/2} \bar{W}_{\bar{h}} I - \eta^{1/2} W_h\|_F^2 \\ =&\|\bar{W}_{\bar{h}} \,\text{concat}([X, \eta^{1/2} I]) - \text{concat}([Y, \eta^{1/2} W_h])\|_F^2 = \|\bar{W}_{\bar{h}} \hat{X} - \hat{Y}\|_F^2. \end{aligned} \tag{10}$$

Theorem 1 from Izenman (1975) states that for the reduced-rank linear regression problem

$$\min_{\bar{W}_{\bar{h}}; \text{rank}(\bar{W}_{\bar{h}}) \le r} \|\bar{W}_{\bar{h}} X - Y\|_F^2, \tag{11}$$

the solution is

$$\bar{W}_{\bar{h}} = \text{SVD}_r(Y X^\dagger X) X^\dagger. \tag{12}$$

Consequently, by change of variables, for

$$\min_{\bar{W}_{\bar{h}}; \text{rank}(\bar{W}_{\bar{h}}) \le r} \|\bar{W}_{\bar{h}} \hat{X} - \hat{Y}\|_F^2, \tag{13}$$

the solution is

$$\bar{W}_{\bar{h}} = \text{SVD}_r((\hat{Y} \hat{X}^\dagger) \hat{X}) \hat{X}^\dagger, \tag{14}$$

which completes the proof.

A.2 Hyperparameters for STRIM+

Compared with STRIM, the only additional hyperparameter in STRIM+ is the parameter ratio used for the key and value projection matrices. To keep the overall parameter ratio fixed, we reduce the ratios of the query and output projection matrices accordingly. Since the query and output projections are often larger than the key and value projections, these reductions are typically smaller than the increase applied to the key and value projections. For Mistral-7B, we set the key/value parameter ratio to 1.0 for total parameter ratios of 0.8, 0.6, and 0.4, and to 0.4 for total parameter ratio 0.2. For the other LLM families, we only evaluate the 0.8 total parameter-ratio setting and use a key/value parameter ratio of 1.0.

A.3 Matched Cross-Model Comparison

Table 5: Six-task classification accuracy across different LLMs at parameter ratio 0.8. SVD-LLM+OPT and STRIM use identical fixed ranks and the same 10-epoch OPT schedule.

Method	Gemma-3-1B	Gemma-3-4B	OPT-6.7B	DeepSeek-7B	Qwen3-8B
	Classification (6-Task Avg. ↑)				
SVD-LLM	34.97	43.09	50.00	49.35	51.16
SVD-LLM+OPT	38.79	47.43	49.99	51.43	52.73
STRIM	**39.11**	**49.22**	**50.05**	**52.50**	**53.78**

A.4 Combination with Pivoting Factorization

For the comparison with PIFA, we separate its mechanism into two parts: a post-compression error-correction step, which also appears in earlier versions of SVD-LLM, and pivoting factorization, which can be applied to any low-rank factorization to further reduce model size. We study these two components by applying them to SVD-LLM and STRIM+.

Comparing SVD-LLM with and without post-compression error correction shows that the correction step improves perplexity but not task accuracy. By contrast, PIFA's advantage over SVD-LLM with error correction indicates that its main benefit comes from pivoting factorization. This mechanism is orthogonal to STRIM: combining STRIM+ with pivoting factorization significantly outperforms PIFA.

Table 6: Comparison with PIFA. STRIM is orthogonal to pivoting factorization, and combining the two yields further gains.

Method	Param Rate 0.8	Param Rate 0.6	Param Rate 0.4	Param Rate 0.2
	WikiText-2 (Perplexity ↓) / Classification (6-Task Avg. ↑)			
PIFA (post-compress corr. & pivot fact.)	5.98 / 56.68	9.64 / 41.50	25.39 / 33.00	**87.25** / 31.39
SVD-LLM	7.62 / 50.57	16.14 / 38.05	**87.25** / 31.55	1062.89 / 31.40
SVD-LLM (+post-compress corr.)	7.33 / 49.41	12.57 / 37.35	29.22 / 32.50	87.25 / 31.36
STRIM+	6.68 / 56.68	9.05 / 49.58	18.29 / 38.72	90.02 / 32.94
STRIM+ (+pivot fact.)	**5.85 / 59.72**	**7.87 / 53.07**	**16.39 / 39.35**	**87.25 / 33.12**

A.5 Instruction Finetuning Results

SVD-LLM proposes applying instruction finetuning after model compression. We follow the same setup and compare against SVD-LLM and AdaSVD using the full Alpaca-cleaned dataset of 51k samples. As shown in Table 7, our method continues to outperform the baselines across parameter ratios under this instruction-finetuning setting.

Table 7: Experimental results with instruction finetuning on the Alpaca-cleaned dataset.

Method	Param Rate 0.8	Param Rate 0.6	Param Rate 0.4	Param Rate 0.2
	WikiText-2 (Perplexity ↓) / Classification (6-Task Avg. ↑)			
SVD-LLM (IFT)	7.99 / 54.06	10.10 / 48.75	15.89 / 40.14	48.18 / 34.61
AdaSVD (IFT)	7.99 / 53.71	10.58 / 49.14	16.14 / 40.90	45.26 / 34.53
STRIM+ (IFT)	**7.74 / 56.15**	**7.99 / 51.95**	**12.37 / 44.26**	**28.33 / 35.62**

A.6 Quantization Results

Table 8: Combining low-rank compression with quantization.

Method	Param Rate 0.8	Param Rate 0.6	Param Rate 0.4	Param Rate 0.2
	WikiText-2 (Perplexity ↓) / Classification (6-Task Avg. ↑)			
SVDLLM (align 128)	7.39 / 52.02	15.40 / 38.68	72.33 / 32.03	938.00 / 31.50
STRIM (align 128)	**6.68 / 56.98**	**9.20 / 48.74**	**17.73 / 39.07**	**81.96 / 32.85**
SVDLLM (align 128, awq)	7.51 / 51.93	15.64 / 38.52	72.33 / 32.16	1096.63 / 31.72
STRIM (align 128, awq)	**6.78 / 56.52**	**9.34 / 48.58**	**18.00 / 38.85**	**84.56 / 32.72**

Table 8 reports the results of combining low-rank compression with 4-bit AWQ quantization. To apply AWQ, the matrix dimensions must be aligned to multiples of a fixed number. Following ASVD (Yuan et al., 2023), we round the ranks up to multiples of 128. The results show that STRIM remains compatible with quantization and can further reduce model size with only a small loss in accuracy.

A.7 Pruning Comparison

We compare STRIM against two pruning baselines, BlockPruner (Zhong et al., 2024) and LLM-Pruner (Ma et al., 2023). Due to the high memory requirements of LLM-Pruner, we evaluate it on a smaller calibration set of 64 samples of length 128. As shown in Table 9, STRIM outperforms both baselines under their respective settings. In particular, STRIM calibrated on the smaller dataset ($S = 64, L = 128$) still surpasses BlockPruner, which uses a much larger calibration set ($S = 256, L = 2048$).

Table 9: Comparison with pruning methods.

Method	PPL / Accuracy
LLM-Pruner (S=64,L=128)	**7.51** / 48.70
STRIM (S=64,L=128)	8.64 / **52.04**
BlockPruner (S=256,L=2048)	9.13 / 50.20
STRIM (S=256,L=2048)	**6.73** / **55.89**

A.8 Optimization Epoch Study

We compare the effect of applying block-MSE optimization to different initializations. As shown in Figure 2, STRIM starts from a substantially stronger initialization and remains better than applying the same optimization directly to SVD-LLM across runs with different optimization budgets in terms of downstream accuracy. At the same time, the right panel shows that the local Rel. RMSE gap shrinks substantially as the optimization budget increases, and even slightly reverses at late epochs, without changing the downstream ranking. This supports the view that the closed-form initialization is an important part of the overall pipeline, rather than the gains arising solely from additional optimization of the local objective.

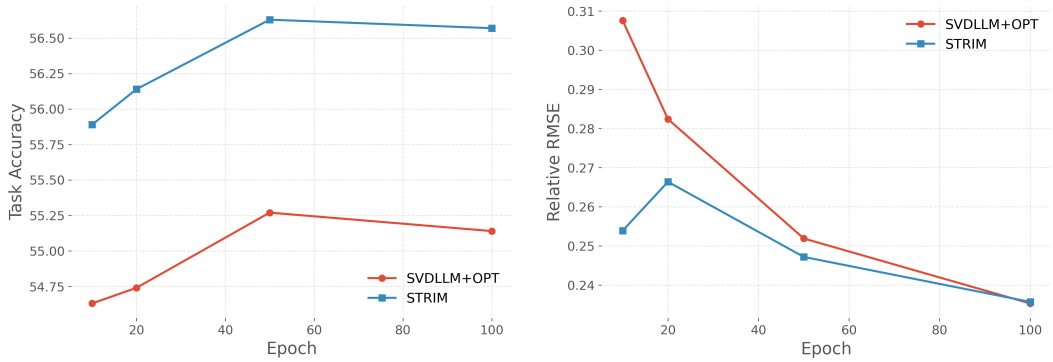

Figure 2: Results for STRIM and SVD-LLM+OPT under different optimization budgets. Left: downstream accuracy at different epoch budgets. Right: local Relative RMSE at the same budgets. Although SVD-LLM+OPT eventually matches or slightly surpasses STRIM on the local objective, the downstream advantage of STRIM persists.

A.9 Regularization Ablation at Parameter Ratio 0.8

We further isolate the role of original-weight regularization at parameter ratio 0.8. Table 10 reports the no-OPT ablation, and Table 11 reports the corresponding with-OPT comparison. Together, these tables show whether the anchored regularizer helps both the initialization quality and the final optimized model. Since vanilla regularization shrinks the solution toward zero rather than toward the original weight, we tune its coefficient on a separate grid instead of reusing the same values as our anchored regularizer.

Table 10: Regularization ablation of STRIM at parameter ratio 0.8 without gradient-based optimization.

Method	PPL / Accuracy
STRIM (no OPT)	**6.94** / **52.80**
STRIM (no OPT, w/o regularization)	7.00 / 51.84
STRIM (no OPT, w/ vanilla regularization)	7.00 / 51.88

Table 11: Regularization ablation of STRIM at parameter ratio 0.8 with gradient-based optimization.

Method	PPL / Accuracy
STRIM (OPT)	**6.73 / 55.89**
STRIM (OPT, w/o regularization)	6.78 / 53.62
STRIM (OPT, w/ vanilla regularization)	6.78 / 53.77

A.10 Layer-wise Relative RMSE

For completeness, we report the layer-wise Relative RMSE for every transformer block of Mistral-7B at parameter ratio 0.8. As shown in Figure 3, STRIM consistently yields lower layer-wise Relative RMSE than the matrix-wise baselines, consistent with the reconstruction objective discussed in the main text.

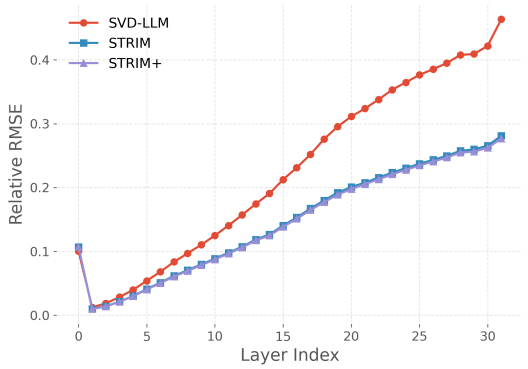

Figure 3: Layer-wise Relative RMSE of Mistral-7B at parameter ratio 0.8.

A.11 GSM8K Examples

We sample 50 GSM8K problems and inspect the outputs of SVD-LLM and STRIM. Table 12 presents five representative examples. The final extracted answer is shown in boldface. In the first three examples, STRIM answers correctly while SVD-LLM fails. In the remaining two, SVD-LLM produces the correct final answer, although its intermediate reasoning is still flawed.

Table 12: Comparison of the outputs of SVD-LLM and STRIM on GSM8K

SVD-LLM	STRIM
Question: Eliza's rate per hour for the first 40 hours she works each week is $10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week? **Answer:** Eliza worked 40 hours at $10 per hour. She worked 5 hours at 1.2 times her regular hourly rate. She worked 40 hours at $10 per hour and 5 hours at 1.2 times her regular hourly rate. She worked 40 hours at $10 per hour and 5 hours at 1.2 times her regular hourly rate. She worked 40 hours at $10 per hour and 5 hours at **1.2** times her regular hourly rate . . . Incorrect	**Question:** Eliza's rate per hour for the first 40 hours she works each week is $10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week? **Answer:** Eliza worked for 40 hours at $10 per hour. She worked for 5 hours at 1.2 times her regular hourly rate. So she worked for 40 hours at $10 per hour and 5 hours at 1.2 times $10 per hour. 40 x $10 = <<40 * 10 = 400 >> 400 dollars for 40 hours at $10 per hour. 5 x 1.2 x $10 = <<5 * 1.2 * 10 = 60 >> 60 dollars for 5 hours at 1.2 times $10 per hour. Altogether, Eliza earned 400 + 60 = <<400 + 60 = 460 >> **460** dollars for the week. ; ; ; ; ; . . . Correct
Question: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs? **Answer:** John takes care of dogs for .5 hours a day. John takes care of dogs for .5 hours a day * 7 days a week = << .5 * 7 = 3.5 >> 3.5 hours a week. John takes care of dogs for .5 hours a day * 7 days a week = << .5 * 7 = 3.5 >> **3.5** hours a week . . . Incorrect	**Question:** John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs? **Answer:** John spends 10 * .5 = <<10 * .5 = 5 >> 5 hours a day taking care of dogs. John takes care of dogs every day, so he spends 5 * 7 = <<5 * 7 = 35 >> **35** hours a week taking care of dogs. ; ; ; ; ; . . . Correct
Question: Grandma Jones baked 5 apple pies for the fireman's luncheon. She cut each pie into 8 pieces and set the five pies out on the buffet table for the guests to serve themselves. At the end of the evening, after the guests had taken and eaten their pieces of pie, there were 14 pieces of pie remaining. How many pieces were taken by the guests? **Answer:** Grandma Jones baked 5 apple pies. She cut each pie into 8 pieces. She set the five pies out on the buffet table for the guests to serve themselves. At the end of the evening, after the guests had taken and eaten their pieces of pie, there were 14 pieces of pie remaining. **14** pieces were taken by the guests. ; ; ; ; ; . . . Incorrect	**Question:** Grandma Jones baked 5 apple pies for the fireman's luncheon. She cut each pie into 8 pieces and set the five pies out on the buffet table for the guests to serve themselves. At the end of the evening, after the guests had taken and eaten their pieces of pie, there were 14 pieces of pie remaining. How many pieces were taken by the guests? **Answer:** Grandma Jones baked 5 apple pies. She cut each pie into 8 pieces. So she cut 40 pieces of pie. She set 5 pies out on the buffet table. So there were 40 pieces of pie on the buffet table. At the end of the evening, after the guests had taken and eaten their pieces of pie, there were 14 pieces of pie remaining. So **26** pieces of pie were taken by the guests. ; ; ; ; ; . . . Correct

Question: Marissa is hiking a 12-mile trail. She took 1 hour to walk the first 4 miles, then another hour to walk the next two miles. If she wants her average speed to be 4 miles per hour, what speed (in miles per hour) does she need to walk the remaining distance? **Answer:** Marissa walked 4 miles in 1 hour. She walked 2 miles in 1 hour. Marissa walked 4 miles + 2 miles = << 4 + 2 = 6 >> 6 miles in 2 hours. Marissa walked 6 miles in 2 hours. Marissa walked **6** miles in . . . Correct, but the reasoning is flawed	**Question:** Marissa is hiking a 12-mile trail. She took 1 hour to walk the first 4 miles, then another hour to walk the next two miles. If she wants her average speed to be 4 miles per hour, what speed (in miles per hour) does she need to walk the remaining distance? **Answer:** Marissa walked 4 miles in 1 hour, so her average speed was 4 miles per hour. She walked another 2 miles in another hour, so her average speed was still 4 miles per hour. She needs to walk the remaining 6 miles in 2 hours so that her average speed will be 4 miles per hour. She needs to walk 6 miles in 2 hours so that her average speed will be **4** miles per hour. . . . Incorrect
Question: Charlie wants to sell beeswax candles. For every pound of beeswax, he can make 10 tapered candles. One pound of beeswax and the wicks cost $10.00 in supplies. If he sells each candle for $2.00 each, what is his net profit if he makes and sells 20 candles? **Answer:** Charlie sells 20 candles. He sells 20 candles for $2.00 each. He sells 20 candles for $2.00 each. He sells **20** cand . . . Correct, but the reasoning is flawed	**Question:** Charlie wants to sell beeswax candles. For every pound of beeswax, he can make 10 tapered candles. One pound of beeswax and the wicks cost $10.00 in supplies. If he sells each candle for $2.00 each, what is his net profit if he makes and sells 20 candles? **Answer:** Charlie can make 10 tapered candles from one pound of beeswax. One pound of beeswax costs $10.00 in supplies. Charlie sells each candle for $2.00 each. Charlie makes and sells 20 candles. Charlie's net profit is $**200.00**.\n ; ; ; ; . . . Incorrect