LOSS AWARE TRUNCATION WITH GRADIENTS FOR LOW-RANK DECOMPOSITION OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The continued scaling of large language models (LLMs) has resulted in rapidly growing memory requirements. Singular Value Decomposition (SVD) provides a principled approach to reducing this burden through low-rank compression. However, existing state-of-the-art SVD-based methods have not fully characterized the relationship between singular components and the performance degradation induced by compression. In this work, we show through the first-order Taylor expansion that the local gradient of each layer provides a direct connection between singular value importance and the resulting loss degradation of the compressed model. Building on this insight, we propose **L**oss-aware **T**runcation with **G**radients (`LTG`), which transforms the weight space according to local loss sensitivity. This transformation redistributes the singular spectrum such that components with greater influence on the model objective are preferentially retained during truncation. We demonstrate the effectiveness of `LTG` over related state-of-the-art approaches across three LLM families and standard benchmarks.

1 INTRODUCTION

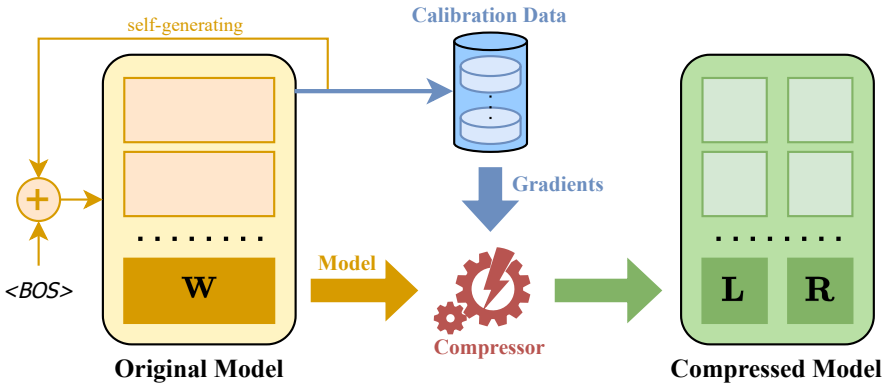


Figure 1: Overview of `LTG`. *<BOS>* denotes the beginning-of-sequence token, $\mathbf{W}$ the original layer weight matrix, and $\mathbf{L}$ and $\mathbf{R}$ the resulting low-rank factors.

The scaling laws of Transformer-based models have driven the remarkable progress in LLMs, while simultaneously driving a substantial increase in model size. As a result, modern LLMs demand substantial memory and computational resources, making inference increasingly expensive. At scale, these requirements translate into high energy consumption and a growing carbon footprint (Thompson et al., 2020; Patterson et al., 2021). At the device level, they can render deployment impractical on resource-constrained platforms. These limitations have motivated extensive research on post-training model compression, which aims to reduce inference costs while preserving the capabilities of the original model (Deng et al., 2020).

However, existing compression methods still face several practical limitations. Quantization relies heavily on hardware support, since the efficiency gains of low-precision or custom numerical for-

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.

mats depend on the availability of compatible hardware and optimized kernels (Frantar et al., 2022; Xiao et al., 2023; Kim et al., 2023). Pruning presents a different trade-off between computational efficiency and model quality. Unstructured pruning can preserve model performance, but its irregular sparsity patterns are difficult to accelerate on parallel hardware (Han et al., 2015; Mishra et al., 2021). In contrast, structured pruning is easier to execute efficiently, but removing entire channels or model components often leads to a substantial performance drop (Frantar & Alistarh, 2023; Ashkboos et al., 2024; Ma et al., 2023). Knowledge distillation requires an additional training stage with a large teacher model, making it less suitable for post-training compression of existing pre-trained LLMs (Hinton et al., 2015; Hou et al., 2020).

Low-rank factorization provides a complementary approach to model compression by approximating a dense weight matrix with the product of two smaller matrices (Golub et al., 1987). This idea is firmly grounded in classical matrix approximation theory and is most commonly realized through SVD, which expresses a matrix as a sum of singular components. Compression is then performed by retaining only the components associated with the largest singular values, yielding the optimal rank-constrained approximation under the Frobenius norm (Klema & Laub, 1980; Eckart & Young, 1936). The resulting factorized layer requires fewer parameters and floating-point operations while preserving the regular dense computation pattern based on standard matrix multiplication kernels (Lebedev et al., 2014). It therefore avoids reliance on sparse accelerators or non-standard numerical formats and can be combined with other compression techniques, including quantization and pruning.

Despite its computational advantages, low-rank factorization remains relatively underexplored for LLM compression. Although truncated SVD is optimal for reconstructing the weight matrix itself, this objective does not account for the propagation of approximation errors to the network output. As a result, components with small singular values may still be functionally important, while larger components may have limited impact on model performance. To address this limitation, several methods incorporate model information into the decomposition process. FWSVD (Hsu et al., 2022) uses Fisher-based sensitivity, ASVD (Yuan et al., 2023) rescales weights using activation statistics, and SVD-LLM (Wang et al., 2025) applies activation whitening to better preserve layer outputs. While these approaches improve upon vanilla SVD, their reliance on surrogate criteria provides only an indirect measure of component importance, which can limit compression effectiveness. Moreover, their dependence on external calibration data can restrict the general applicability of the resulting compression framework.

In this work, we introduce `LTG` (Fig. 1), a simple self-calibrated, loss-aware post-training compression method that establishes a direct relationship between weight truncation and the model objective. We first show that the importance of each singular component depends not only on its singular value, but also on its interaction with the local gradient. Based on this observation, we derive a gradient-informed compression objective and use it to redistribute the singular components before truncation. This allows `LTG` to preserve the directions that are most important to the model objective during compression.

Our main contributions are summarized as follows:

- We provide mathematical grounding showing that the importance of singular components is determined not only by the geometry of the weight matrix, but also by the interaction of the corresponding weight directions with the local loss landscape (Sec. 3.2).
- Building on this foundation, we propose `LTG`, a self-calibrated gradient-aware compression method that preserves loss sensitive directions during weight truncation (Sec. 3.2).
- We conduct extensive experiments across multiple models, datasets, and compression ratios to validate the effectiveness and robustness of `LTG` (Sec. 4).

2 Related Works

Modern LLMs often contain billions of parameters, making training-aware compression methods prohibitively expensive. This has motivated the development of post-training compression techniques, which reduce model size and computational cost without requiring full retraining. Existing approaches can be broadly categorized into four groups: unstructured pruning, structured pruning, quantization, and low-rank approximation.

Pruning and Quantization. Unstructured pruning removes individual weights by setting them to zero while preserving the original tensor dimensions (Frantar & Alistarh, 2023). Although this approach can achieve high sparsity, the resulting irregular patterns are often difficult to exploit efficiently on standard hardware, thereby limiting practical reductions in latency and memory usage. Structured pruning instead removes entire channels, neurons, or other regular model components. This produces hardware-friendly architectures with reduced dense computation, but often incurs substantial performance degradation, particularly at high compression ratios (Ashkboos et al., 2024; Ma et al., 2023; Zhong et al., 2025).
Quantization reduces the storage and computational requirements of LLMs by representing weights and activations using lower-precision numerical formats (Xiao et al., 2023). However, the achievable inference speedup depends strongly on hardware support and the availability of efficient low-precision kernels (Yao et al., 2022).

Low-rank Approximation. This approach is commonly implemented through SVD, provides an alternative approach to compressing LLMs. Unlike the methods discussed above, SVD-based approaches replace each weight matrix with two smaller factors and can therefore be deployed using standard matrix multiplication kernels without specialized hardware support (Ioannou et al., 2015; Lebedev et al., 2014). Early methods applied truncated SVD directly to model weights, but often suffered substantial accuracy degradation because the decomposition considered only the geometry of the weight matrix and ignored its interaction with other components (Chen et al., 2021).
FWSVD (Hsu et al., 2022) addresses this limitation by using Fisher information to estimate the importance of individual parameters. ASVD (Yuan et al., 2023) further improve this estimation by applies a diagonal transformation derived from activation statistics to rescale each weight matrix before decomposition, thereby balancing the contributions of different input channels. While ASVD improves upon vanilla SVD, its singular-value truncation criterion is not directly optimized for the layer-wise reconstruction error induced by compression.
SVD-LLM (Wang et al., 2025) bridges this gap through truncation-aware data whitening. By applying a whitened transformation derived from the input covariance matrix, it reformulates the layer-output reconstruction error as the sum of the squared discarded singular values. In the transformed space, retaining the largest singular components is therefore optimal for minimizing the reconstruction error of each individual layer. However, this does not necessarily preserve the model objective, as perturbations introduced at one layer subsequently propagate through downstream layers, nonlinearities, and other architectural components.

In this work, we show that the effect of weight perturbations introduced by compression at a given layer should depend on their interaction with the corresponding local loss gradient. Motivated by this observation, we propose `LTG`, a loss-aware low-rank compression method that explicitly incorporates local sensitivity into the decomposition process. By weighting reconstruction errors according to their influence on the objective, `LTG` more directly aligns low-rank truncation with the resulting change in model loss.

For the remainder of this paper, we focus exclusively on low-rank approximation approaches. Pruning and quantization are not direct baselines, as they are complementary and can be combined with low-rank approximation to obtain additional benefits. Further comparisons with these methods can be found in Wang et al. (2025) and Appendix C.

3 METHOD

Consider a large language model $\mathcal{M}(\cdot)$ comprising L linear layers. Given an input sequence $\mathbf{X} \in \mathbb{R}^{N\times D}$ of N tokens with D-dimensional representations, the model produces an output token $\mathbf{Y} \in \mathbb{R}^{1\times D}$. For each linear layer l, let $\mathbf{X}_l \in \mathbb{R}^{N\times I_l}$ and $\mathbf{W}_l \in \mathbb{R}^{O_l\times I_l}$ denote its input activation and weight matrix, respectively. We define $\mathcal{F}_l$ as the composite transformation from the layer output to the final training loss. The loss can therefore be written from the perspective of layer l as

$$\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y}) \triangleq \mathcal{F}_l\left(\mathbf{X}_l \mathbf{W}_l^\top\right) \in \mathbb{R}, \qquad l = 0, \ldots, L-1, \tag{1}$$

where $\hat{\mathbf{Y}}$ denotes the ground-truth next token. For notational simplicity, we omit the layer index l whenever it is clear from context.

3.1 Model Compression with Low-Rank Approximation

During inference, a layer's weight $\mathbf{W}$ is often a dominant contributor to both memory footprint and computational cost. A standard way to reduce these costs is to replace $\mathbf{W}$ with a rank-r approximation $\widetilde{\mathbf{W}}$, with $r \ll \min(O, I)$, obtained by solving

$$\min_{\text{rank}(\widetilde{\mathbf{W}}) \leq r} \left\| \mathbf{W} - \widetilde{\mathbf{W}} \right\|_F^2. \tag{2}$$

Applying SVD to $\mathbf{W}$ and retaining the components indexed by the set $\mathcal{R}$, with $|\mathcal{R}| = r$, gives

$$\widetilde{\mathbf{W}} = \text{Trunc}_{\mathcal{R}}(\mathbf{W}) = \sum_{i \in \mathcal{R}} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top, \tag{3}$$

where σ_i, $\mathbf{u}_i$, and $\mathbf{v}_i$ denote the i-th singular value and its corresponding left and right singular vectors, respectively. By the Eckart-Young theorem (Eckart & Young, 1936), $\widetilde{\mathbf{W}}$ is the optimal solution of Eq. 2 when $\mathcal{R}$ contains the indices of the r largest singular values σ_i.

Equivalently, we can absorb the singular values symmetrically into two low-rank factors:

$$\mathbf{L} = [\sqrt{\sigma_i}\mathbf{u}_i]_{i \in \mathcal{R}}, \qquad \mathbf{R} = [\sqrt{\sigma_i}\mathbf{v}_i]_{i \in \mathcal{R}}^\top. \tag{4}$$

At inference time, the original projection $\mathbf{X}\mathbf{W}^\top$ is replaced by two consecutive smaller projections using $\mathbf{L} \in \mathbb{R}^{O \times r}$ and $\mathbf{R} \in \mathbb{R}^{r \times I}$. This reduces the storage cost from $\mathcal{O}_{\text{space}}(OI)$ to $\mathcal{O}_{\text{space}}(r(O+I))$, and the computational cost from $\mathcal{O}(NOI)$ to $\mathcal{O}(Nr(I+O))$.

3.2 Loss-Aware Truncation with Gradients

Recent studies have shown that the weight matrix $\mathbf{W}$ is not necessarily low-rank. Consequently, directly applying truncated SVD to $\mathbf{W}$ can substantially degrade model performance, since the singular components that best reconstruct the weight matrix are not necessarily those most important for preserving the model objective (Chen et al., 2021; Hsu et al., 2022; Yuan et al., 2023; Wang et al., 2025). This suggests that low-rank components should instead be selected according to their influence on the model loss.

Loss-aware Truncation Objective. To explicitly account for the effect of compression on the model objective, we seek a rank-r approximation $\widetilde{\mathbf{W}}$ that minimizes the discrepancy between the losses induced by the original and compressed weights:

$$\Delta\mathcal{L}^2 = \left[\mathcal{F}(\mathbf{X}\mathbf{W}^\top) - \mathcal{F}(\mathbf{X}\widetilde{\mathbf{W}}^\top) \right]^2. \tag{5}$$

Directly optimizing Eq. 5 is challenging, as $\mathcal{F}$ contains all subsequent transformations between the current layer and the final loss. We therefore characterize the local effect of weight perturbation using a first-order approximation.[1]

Lemma 1 (First-order loss perturbation)**.** *Let $\widetilde{\mathbf{W}} = \mathbf{W} + \Delta\mathbf{W}$, and assume that $\mathcal{F}$ is differentiable at $\mathbf{X}\mathbf{W}^\top$. Then*

$$\mathcal{F}(\mathbf{X}\widetilde{\mathbf{W}}^\top) \approx \mathcal{F}(\mathbf{X}\mathbf{W}^\top) + \left\langle \frac{\partial \mathcal{L}}{\partial(\mathbf{X}\mathbf{W}^\top)}, \mathbf{X}(\widetilde{\mathbf{W}} - \mathbf{W})^\top \right\rangle_F, \tag{6}$$

where $\langle ., . \rangle_F$ denotes the Frobenius inner product. The proof is provided in Appendix A.

Applying Lemma 1 to Eq. 5 yields

$$\begin{aligned} \Delta\mathcal{L}^2 &\approx \left\langle \frac{\partial \mathcal{L}}{\partial(\mathbf{X}\mathbf{W}^\top)}, \mathbf{X}(\mathbf{W} - \widetilde{\mathbf{W}})^\top \right\rangle_F^2 \\ &= \left\langle \mathbf{W} - \widetilde{\mathbf{W}}, \mathbf{G}_{\mathbf{W}} \right\rangle_F^2, \end{aligned} \tag{7}$$

[1] We restrict the analysis to first order for computational efficiency. Further discussion is provided in Appendix A.1.

where $\mathbf{G}_{\mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \in \mathbb{R}^{O \times I}$ denotes the local loss gradient with respect to the layer weight.

Eq. 7 shows that the effect of a compression perturbation depends not only on its magnitude, but also on its alignment with the local loss gradient. Consequently, similar perturbations in weight can induce substantially different changes in the model objective depending on their directions (verified in Sec. 4.2).

However, directly minimizing Eq. 7 is undesirable, since any perturbation orthogonal to $\mathbf{G}_{\mathbf{W}}$ produces zero first-order loss change regardless of its magnitude. We therefore consider the following Cauchy-Schwarz upper bound (Horn & Johnson, 2012):

$$\begin{aligned} \left\langle \mathbf{W} - \widetilde{\mathbf{W}}, \mathbf{G}_{\mathbf{W}} \right\rangle_F^2 &= \mathrm{Tr}\Big[(\mathbf{W} - \widetilde{\mathbf{W}})\mathbf{G}_{\mathbf{W}}^\top\Big]^2 \\ &\leq O \left\| (\mathbf{W} - \widetilde{\mathbf{W}})\mathbf{G}_{\mathbf{W}}^\top \right\|_F^2 . \end{aligned} \tag{8}$$

Dropping the constant factor O, we obtain the gradient-aware compression objective

$$\mathcal{G}(\widetilde{\mathbf{W}}, r) \triangleq \left\| (\mathbf{W} - \widetilde{\mathbf{W}})\mathbf{G}_{\mathbf{W}}^\top \right\|_F^2 . \tag{9}$$

Gradient Whitening. Eq. 9 suggests that the approximation error should be measured in a geometry induced by the local gradient rather than directly in the original weight space. To obtain an SVD-compatible form, we define

$$\mathbf{G}\mathbf{G}^\top = \mathbf{G}_{\mathbf{W}}^\top \mathbf{G}_{\mathbf{W}}, \tag{10}$$

where $\mathbf{G} \in \mathbb{R}^{I \times I}$ is obtained through Cholesky factorization.[2]

We then parameterize the compressed weight as

$$\widetilde{\mathbf{W}} \triangleq \mathrm{Trunc}_{\mathcal{R}}(\mathbf{W}\mathbf{G})\mathbf{G}^{-1}, \tag{11}$$

where $\mathcal{R}$ denotes the set of retained singular components. Let

$$\mathbf{W}\mathbf{G} = \sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^\top . \tag{12}$$

Using the residual form of truncated SVD (detailed derivation in Appendix A.1), Eq. 9 becomes

$$\begin{aligned} \mathcal{G}(\widetilde{\mathbf{W}}, r) &= \left\| \left[\mathbf{W}\mathbf{G}\mathbf{G}^{-1} - \mathrm{Trunc}_{\mathcal{R}}(\mathbf{W}\mathbf{G})\mathbf{G}^{-1}\right] \mathbf{G}_{\mathbf{W}}^\top \right\|_F^2 \\ &= \sum_{i \notin \mathcal{R}} \left\| \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \mathbf{G}^{-1} \mathbf{G}_{\mathbf{W}}^\top \right\|_F^2 \\ &= \sum_{i \notin \mathcal{R}} \sigma_i^2 \, \mathrm{Tr}\big[\mathbf{u}_i \mathbf{v}_i^\top \mathbf{G}^{-1} \mathbf{G}_{\mathbf{W}}^\top \mathbf{G}_{\mathbf{W}} (\mathbf{G}^{-1})^\top \mathbf{v}_i \mathbf{u}_i^\top\big] \\ &= \sum_{i \notin \mathcal{R}} \sigma_i^2 . \end{aligned} \tag{13}$$

The third equality follows from $\|\mathbf{A}\|_F^2 = \mathrm{Tr}(\mathbf{A}\mathbf{A}^\top)$. For the final equality, Eq. 10 gives $\mathbf{G}^{-1}\mathbf{G}_{\mathbf{W}}^\top \mathbf{G}_{\mathbf{W}} (\mathbf{G}^{-1})^\top = \mathbf{I}$, while the orthonormality of the singular vectors gives $\mathbf{v}_i^\top \mathbf{v}_i = \mathrm{Tr}(\mathbf{u}_i \mathbf{u}_i^\top) = 1$.

Eq. 13 restores the standard truncated SVD form, but in the gradient-transformed space. Therefore, minimizing the loss-aware objective $\mathcal{G}(\widetilde{\mathbf{W}}, r)$ is equivalent to retaining the r largest singular values of $\mathbf{W}\mathbf{G}$.

The corresponding low-rank factors are

$$\mathbf{L} = [\sqrt{\sigma_i}\mathbf{u}_i]_{i \in \mathcal{R}} , \qquad \mathbf{R} = [\sqrt{\sigma_i}\mathbf{v}_i]_{i \in \mathcal{R}}^\top \mathbf{G}^{-1}. \tag{14}$$

Unlike vanilla SVD, which ranks singular components solely according to the geometry of $\mathbf{W}$, `LTG` performs truncation in a gradient-transformed space, allowing the resulting singular spectrum to reflect the local sensitivity of the model objective.

[2] Any alternative factorization that satisfies the same matrix relation is applicable. We adopt Cholesky factorization for computational efficiency, while observing no considerable difference in the final performance among these choices. Further details are provided in Appendix B.

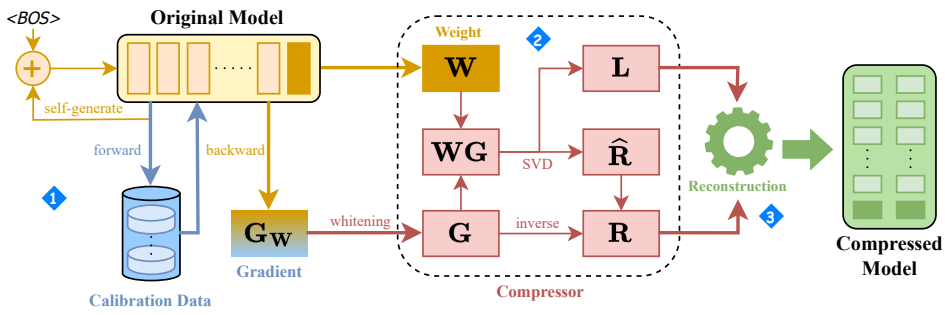


Figure 2: Detailed pipeline of LTG, consisting of three stages: (1) self-calibration phase that generates calibration data and computes the corresponding gradients, (2) loss-aware low-rank compression phase that transforms and factorizes each layer weight, and (3) reconstruction phase that adapts the compressed model to the resulting low-rank representation.

Data-free Calibration Phase. We adopt a calibration phase to estimate $\mathbf{G}$ for guiding the compression process. Prior studies (Williams et al., 2025; Ji et al., 2025) have shown that the choice of calibration data can substantially affect post-training compression, with data that better reflect the pretraining distribution generally leading to better preservation of downstream performance. For modern LLMs, public access to such data is often unavailable. To remove this dependency, we generate calibration sequences directly from the pretrained model itself.
Specifically, we generate each calibration sequence $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_N\}$ of length N by initializing $\mathbf{s}_1$ as the beginning-of-sequence token. Each subsequent token is sampled autoregressively from the model's predictive distribution:

$$\mathbf{s}_n \sim \mathrm{Softmax}(\mathcal{M}(\mathbf{S}_{1:n-1}))\,, \qquad n = 2, \ldots, N. \tag{15}$$

The resulting self-generated sequences are then used to compute the local gradients $\mathbf{G}_\mathbf{W}$ required to estimate $\mathbf{G}$.

Reconstruction Phase. Although our truncation strategy preserves the components most relevant to the model objective, compression still introduces information loss, particularly at high compression ratios. We therefore adopt the LoRA-based reconstruction strategy of Wang et al. (2025) to recover the resulting performance degradation. By optimizing only low-rank parameters on a small reconstruction dataset, this phase allows the compressed model to efficiently adapt to its new representation without full model retraining.

An overview of the method is described in Fig. 2.

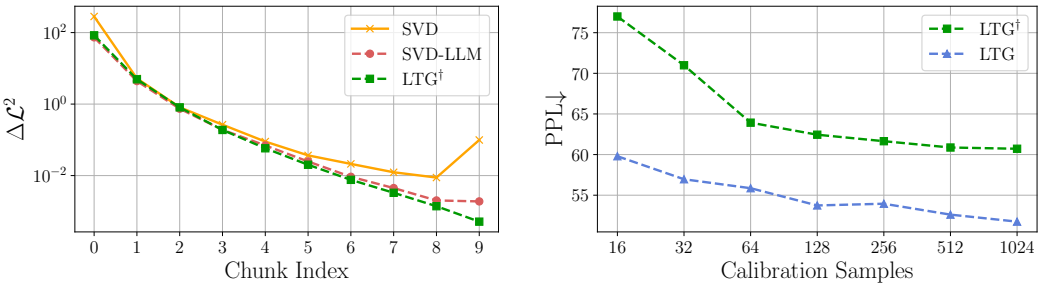
4 Experimental Results

In this section, we present experiments designed to demonstrate the effectiveness of LTG. We begin by outlining the experimental setup in Sec. 4.1. Then, in Sec. 4.2, we conduct experiments to validate the assumptions introduced in Sec. 3.2. Sec. 4.3 compares LTG with various related state-of-the-art methods across multiple models and datasets.

4.1 Experimental Setup

Baselines. Our primary focus is to investigate the effectiveness of low-rank factorization for LLMs compression. Accordingly, we compare LTG against representative state-of-the-art SVD-based compression methods, including FWSVD, ASVD, SVD-LLM, and vanilla SVD. Other compression paradigms, such as pruning and quantization, are orthogonal to our focus and are therefore excluded from our main evaluation.

Models and Datasets. We conduct all ablation studies on OPT-350M (Zhang et al., 2022) for computational efficiency. To assess the generalizability of LTG, we evaluate it across three model families comprising five models: OPT-6.7B (Zhang et al., 2022), LLaMA-7B (Touvron et al., 2023a),



(a) (b)

Figure 3: **(a)** Loss perturbation induced by removing equally sized singular-component chunks ordered by decreasing magnitude. **(b)** Effect of calibration set size on C4 perplexity for self-generated (LTG) and external calibration (LTG†).

LLaMA2-7B (Touvron et al., 2023b), Vicuna-7B (Chiang et al., 2023), and Mistral-7B (Jiang et al., 2023). For a fair comparison with prior state-of-the-art methods, we follow the same evaluation protocol and consider eight datasets: C4 (Raffel et al., 2020) for language modeling; OpenBookQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-Easy (Clark et al., 2018), PIQA (Bisk et al., 2020), and MathQA (Amini et al., 2019) for classification; and GSM8K (Cobbe et al., 2021) for generation. All downstream evaluations are conducted using the Language Model Evaluation Harness framework (Gao et al., 2023).

Metrics. We report perplexity on C4, accuracy on the six classification datasets, and exact-match accuracy on GSM8K. Lower perplexity and higher accuracy indicate better performance.

Experimental Setup. To ensure a fair comparison, we use the same fixed random seed across all experiments and set the sequence length to 2048. Following other baselines, we use a uniform compression ratio to determine each layer rank and adopt WikiText-2 (Merity et al., 2017) whenever external calibration data are required. All experiments are implemented in PyTorch 2.0.1 and conducted on a single NVIDIA L40S GPU with 48 GB of VRAM. For the reconstruction phase, we follow the hyperparameter configuration of SVD-LLM (Wang et al., 2025). Specifically, we finetune the compressed model with LoRA adapters on 50K samples from the Alpaca dataset (Taori et al., 2023), following the reconstruction strategy adopted in LLM-Pruner (Ma et al., 2023).

Further results, including detailed numerical comparisons, more comprehensive ablation studies, evaluations across a broader range of experimental settings, and robustness analyses over multiple random seeds, are provided in Appendix C.

4.2 Preliminary Results

In this section, we empirically examine whether the gradient-aware criterion derived in Section 3.2 better captures the effect of low-rank perturbations on the model objective, and analyze the sensitivity of LTG to its calibration budget. For convenience, we denote by LTG† the variant that uses external calibration data. Unless otherwise stated, all models are evaluated before reconstruction.

Validating Loss-aware Component Importance. Our theoretical analysis in Sec. 3.2 shows that perturbations of comparable magnitude can induce substantially different changes in the model objective, depending on their alignment with the local loss gradient (see Eq. 7). We empirically examine this property in Fig. 3a.
Directly imposing identical perturbation magnitudes across different methods is impractical, as each method applies a distinct transformation that rescales the weight space before decomposition. We therefore partition the singular components of each weight matrix in OPT-350M into 10 equally sized chunks, ordered in descending singular value magnitude, such that lower indexed chunks contain components with larger singular values. We independently remove each chunk while keeping the number of discarded components fixed, and measure the resulting squared loss change $\Delta\mathcal{L}^2$

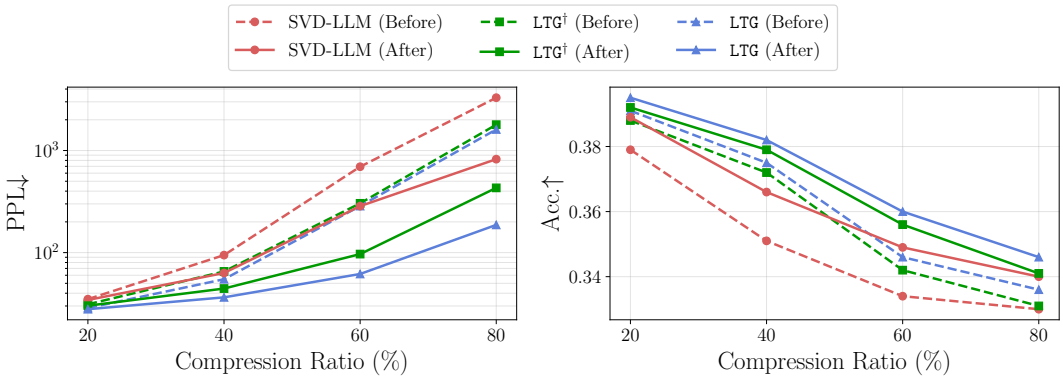


Figure 4: Performance of OPT-350M compressed by different methods across varying compression ratios, before and after reconstruction. The left plot reports C4 perplexity (PPL↓), while the right plot reports average accuracy (Acc.↑) across the six downstream classification datasets. Dashed and solid lines denote results before and after reconstruction, respectively.

with respect to the original model during inference on C4. We compare vanilla SVD as the standard weight-only baseline, SVD-LLM as a representative activation-aware method due to its strong performance among existing SVD-based approaches, and LTG† for a fair comparison.
As shown in Fig. 3a, LTG† tends to induce smaller $\Delta\mathcal{L}^2$ than competing approaches as the chunk magnitude decreases, particularly for components ranked as less important by each method. These results support our central hypothesis that singular-component importance should not be determined solely by the geometry of the weight perturbation or by its interaction with input activations. Instead, it should be characterized by how strongly the corresponding perturbation interacts with the local loss gradient. By incorporating this gradient information through the proposed transformation, LTG provides a more faithful ordering of singular components according to their impact on the model objective.

Effect of Calibration Set Size. We next study the sensitivity of LTG to the number of calibration samples. Fig. 3b reports the perplexity of the OPT-350M model at 40% compression ratio as the calibration set size increases from 16 to 1024 samples. For both self-generated calibration (LTG) and external calibration (LTG†), performance improves substantially at small calibration budgets and gradually saturates as more samples are introduced.
The gains beyond 256 samples are marginal, indicating that the gradient statistics required by LTG can be estimated reliably from a relatively small calibration set. We therefore use 256 calibration samples in all subsequent experiments, consistent with the configuration adopted in prior methods.

Effectiveness of LTG Design. We further evaluate the effectiveness of the proposed design by comparing LTG with SVD-LLM, both before and after reconstruction. As shown in Fig. 4, LTG consistently achieves lower perplexity and higher average downstream accuracy than SVD-LLM across all compression ratios. Importantly, this advantage is preserved after reconstruction, indicating that the benefit of the proposed gradient-aware truncation is complementary to the subsequent reconstruction stage rather than being compensated for by it.
We also compare the proposed self-calibration strategy with external calibration variant LTG†. Across all compression ratios, LTG consistently outperforms LTG† in both perplexity and average downstream accuracy. This result, together with Fig. 3b, shows that self-generated samples provide sufficiently informative gradient statistics for guiding compression and can even outperform external calibration data. These findings demonstrate that LTG improves upon activation-aware compression while eliminating the dependence on an external calibration corpus.

4.3 Main Results

We next evaluate the generalizability of LTG across five LLMs from three model families under compression ratios ranging from 20% to 80%. Table 1 compares LTG with vanilla SVD, FWSVD,

Table 1: Performance of different LLMs across varying compression ratios. We report C4 perplexity (PPL↓), average downstream accuracy (Acc.↑), and GSM8K exact-match accuracy (GSM8K↑). The best and second-best results under each compression ratio are highlighted in bold and underlined.

Ratio (Mem.)	Method	OPT-6.7B			Vicuna-7B			LLaMA 2-7B			LLaMA-7B			Mistral-7B		
		PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑
0% (∼13GB)	Original	12.00	0.551	0.018	8.82	0.608	0.107	7.07	0.617	0.054	7.12	0.617	0.042	7.83	0.661	0.065
20% (∼10GB)	SVD	107074	0.340	0.000	24740	0.328	0.000	28732	0.332	0.001	18735	0.342	0.005	91275	0.335	0.001
	FWSVD	38.73	0.489	0.018	18066	0.326	0.000	10207	0.325	0.000	15481	0.327	0.000	15468	0.328	0.000
	ASVD	18.37	0.506	0.021	260.50	0.393	0.008	98.18	0.398	0.012	37.62	0.442	0.011	19.94	0.545	0.025
	SVD-LLM	15.78	**0.541**	0.022	12.45	0.557	0.040	11.55	0.567	0.036	11.41	0.564	0.032	13.50	0.581	0.067
	LTG	**14.22**	0.540	**0.025**	**12.14**	**0.564**	**0.056**	**10.34**	**0.570**	**0.037**	**10.19**	**0.576**	**0.036**	**10.83**	**0.615**	**0.077**
40% (∼8GB)	SVD	23059	0.336	0.003	53218	0.330	0.000	56611	0.321	0.000	46053	0.338	0.000	68454	0.329	0.000
	FWSVD	1544	0.372	0.002	80595	0.331	0.000	137506	0.332	0.000	149310	0.331	0.000	40084	0.329	0.000
	ASVD	82.97	0.350	0.006	5144	0.336	0.000	3769.95	0.326	0.000	10551	0.336	0.000	2510	0.356	0.007
	SVD-LLM	20.61	0.507	**0.025**	16.43	0.516	0.032	15.53	**0.515**	0.029	14.72	0.511	0.020	18.91	0.523	0.033
	LTG	**17.34**	**0.514**	0.022	**15.97**	**0.517**	**0.033**	**13.59**	0.514	**0.030**	**13.06**	**0.516**	**0.021**	**14.46**	**0.535**	**0.034**
60% (∼5GB)	SVD	4612	0.340	0.000	73880	0.331	0.000	71181	0.329	0.000	107038	0.330	0.000	34641	0.328	0.000
	FWSVD	17843	0.331	0.000	53995	0.327	0.000	74041	0.336	0.000	107134	0.330	0.000	151761	0.330	0.000
	ASVD	6459	0.330	0.002	44663	0.335	0.000	32526	0.337	0.000	44323	0.324	0.000	21873	0.334	0.000
	SVD-LLM	49.83	0.421	0.012	26.01	0.422	0.024	25.13	**0.442**	0.014	24.26	0.440	0.024	30.75	0.431	0.021
	LTG	**28.90**	**0.473**	**0.021**	**24.97**	**0.439**	**0.026**	**21.11**	0.433	**0.020**	**19.44**	**0.441**	**0.026**	**23.70**	**0.442**	**0.023**
80% (∼3GB)	SVD	4764	0.334	0.000	63783	0.319	0.000	53468	0.325	0.000	216297	0.334	0.000	15211	0.326	0.000
	FWSVD	74437	0.333	0.002	38064	0.325	0.000	36990	0.334	0.000	298022	0.327	0.000	12934	0.326	0.000
	ASVD	60676	0.338	0.000	23450	0.336	0.000	20303	0.327	0.000	141347	0.327	0.000	19657	0.333	0.000
	SVD-LLM	1538	0.344	0.000	60.00	0.363	0.014	59.51	0.374	0.014	60.89	0.342	0.004	66.45	0.374	0.016
	LTG	**706.2**	**0.358**	**0.006**	**53.62**	**0.371**	**0.021**	**46.41**	**0.376**	**0.020**	**42.06**	**0.373**	**0.017**	**51.11**	**0.376**	**0.018**

ASVD, and SVD-LLM on language modeling, downstream classification, and mathematical reasoning tasks.

Overall Performance. As shown in Table 1, LTG consistently achieves the lowest perplexity among all compressed models across every model and compression ratio. In contrast, vanilla SVD, FWSVD, and ASVD often suffer substantial performance degradation as the compression ratio increases, indicating that weight magnitude, Fisher information, or activation statistics alone are insufficient to reliably identify functionally important low-rank components. SVD-LLM remains substantially more competitive by explicitly preserving layer-wise activation reconstruction, yet LTG further improves upon it by incorporating the influence of each perturbation on the model objective. The advantage of LTG is also reflected in downstream performance. Across most settings, LTG achieves higher average accuracy and GSM8K exact-match accuracy than SVD-LLM, while occasional differences are marginal. These results show that the our compression objective achieved by incorporating local gradient information to the factorization process generally translates to better preservation of downstream capabilities.

Robustness under Aggressive Compression. The benefit of LTG becomes particularly pronounced at higher compression ratios. At 60% and 80% compression, conventional SVD-based approaches exhibit severe degradation across nearly all models, whereas LTG remains substantially more robust. Even relative to SVD-LLM, LTG consistently yields lower perplexity under these aggressive settings and generally preserves stronger downstream performance. This behavior supports our central motivation: as more singular components must be discarded, selecting components according to their influence on the model objective becomes increasingly important.

5 Conclusion

We presented LTG, a self-calibrated, gradient-aware compression framework for LLM. LTG first derives a loss-aware criterion that directly relates weight truncation to its impact on the model objective, and then performs SVD in a gradient-transformed space to prioritize singular components according to their functional importance. The required gradient statistics are obtained from self-generated calibration data, eliminating the dependence on external calibration datasets.

Across multiple LLM families, compression ratios, and downstream tasks, LTG consistently achieves a stronger compression-performance trade-off than existing SVD-based methods, with increasingly clear advantages under aggressive compression. These results highlight local loss sensitivity as an effective signal for identifying important low-rank components and suggest a principled direction for objective-aware model compression.

AI USE STATEMENT

In this work, we used generative AI tools for assisting in the providing feedback on research experiments, implementing methods, and assisting with translation. We have not used generative AI tools for the rest of the required disclosure tasks. Additionally, we used generative AI tools for creating or editing software code, sourcing/searching for information, and editing a research paper to improve readability. We have reviewed all AI-assisted work. All AI-assisted code and any manuscript sections edited with the help of generative AI were reviewed and verified by at least two authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The complete experimental setup, including the models, datasets, and evaluation environment, is detailed in Sec. 4.1. Further implementation details are provided in Appendix B. The code will be released publicly upon acceptance.

REFERENCES

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, 2019.

Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, volume 2024, pp. 11682–11701, 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*, 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.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

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, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4): 485–532, 2020.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

Gene H Golub, Alan Hoffman, and Gilbert W Stewart. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its applications*, 88:317–327, 1987.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33: 9782–9793, 2020.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. *arXiv preprint arXiv:2207.00112*, 2022.

Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*, 2015.

Yixin Ji, Yang Xiang, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. Beware of calibration data for pruning large language models. In *International Conference on Learning Representations*, volume 2025, pp. 18604–18620, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

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.

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 *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

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.

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.

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, 2023.

Neil C Thompson, Kristjan Greenewald, Keeheon Lee, Gabriel F Manso, et al. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 10(2), 2020.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023a. URL https://arxiv.org/abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

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*, 2025. URL https://openreview.net/forum?id=LNYIUouhdt.

Miles Williams, George Chrysostomou, and Nikolaos Aletras. Self-calibration for language model quantization and pruning. 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. 10149–10167, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183, 2022.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, 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? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 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.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. Blockpruner: Fine-grained pruning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5065–5080, 2025.

A Additional Theoretical Details

A.1 Additional Lemmas

Lemma 2 (Perturbation bound). *For any $\mathbf{X} \in \mathbb{R}^{N\times I}$ and $\Delta\mathbf{W} \in \mathbb{R}^{O\times I}$,*

$$\left\|\mathbf{X}\Delta\mathbf{W}^\top\right\|_F \le \|\mathbf{X}\|_2\|\Delta\mathbf{W}\|_F. \tag{16}$$

Proof. Let $\Delta\mathbf{W}^\top = [\mathbf{w}_1, \dots, \mathbf{w}_O]$, where $\mathbf{w}_i \in \mathbb{R}^I$. Then

$$\mathbf{X}\Delta\mathbf{W}^\top = [\mathbf{X}\mathbf{w}_1, \dots, \mathbf{X}\mathbf{w}_O].$$

By the definition of the spectral norm,

$$\|\mathbf{X}\mathbf{w}_i\|_2 \le \|\mathbf{X}\|_2\|\mathbf{w}_i\|_2.$$

Therefore,

$$\left\|\mathbf{X}\Delta\mathbf{W}^\top\right\|_F^2 = \sum_{i=1}^{O} \|\mathbf{X}\mathbf{w}_i\|_2^2 \tag{17}$$

$$\le \|\mathbf{X}\|_2^2 \sum_{i=1}^{O} \|\mathbf{w}_i\|_2^2 \tag{18}$$

$$= \|\mathbf{X}\|_2^2\|\Delta\mathbf{W}\|_F^2. \tag{19}$$

Taking square roots on both sides completes the proof.

Proof of Lemma 1. Since $\widetilde{\mathbf{W}} = \mathbf{W} + \Delta\mathbf{W}$, we have

$$\mathbf{X}\widetilde{\mathbf{W}}^\top = \mathbf{X}\mathbf{W}^\top + \mathbf{X}\Delta\mathbf{W}^\top. \tag{20}$$

Applying the first-order Taylor expansion of $\mathcal{F}$ around $\mathbf{X}\mathbf{W}^\top$ gives

$$\mathcal{F}(\mathbf{X}\widetilde{\mathbf{W}}^\top) = \mathcal{F}(\mathbf{X}\mathbf{W}^\top) + \left\langle \frac{\partial\mathcal{L}}{\partial(\mathbf{X}\mathbf{W}^\top)}, \mathbf{X}\Delta\mathbf{W}^\top \right\rangle_F + o\big(\|\mathbf{X}\Delta\mathbf{W}^\top\|_F\big). \tag{21}$$

From Lemma 2,

$$\|\mathbf{X}\Delta\mathbf{W}^\top\|_F \le \|\mathbf{X}\|_2\|\Delta\mathbf{W}\|_F.$$

Hence, for a fixed input activation $\mathbf{X}$, a sufficiently small weight perturbation $\|\Delta\mathbf{W}\|_F$ also induces a small perturbation in the layer output. Neglecting the higher-order term yields

$$\mathcal{F}(\mathbf{X}\widetilde{\mathbf{W}}^\top) \approx \mathcal{F}(\mathbf{X}\mathbf{W}^\top) + \left\langle \frac{\partial\mathcal{L}}{\partial(\mathbf{X}\mathbf{W}^\top)}, \mathbf{X}(\widetilde{\mathbf{W}} - \mathbf{W})^\top \right\rangle_F, \tag{22}$$

which proves the result.

Discussion on Higher-Order Extensions. We restrict the analysis to the first-order term, as explicitly incorporating higher-order terms would require estimating curvature information for each compressed layer, which is computationally prohibitive at the scale of LLMs. In particular, a second-order expansion involves $\frac{1}{2}\,\mathrm{vec}(\Delta\mathbf{Z})^\top\mathbf{H}_\mathbf{Z}\,\mathrm{vec}(\Delta\mathbf{Z})$, where $\Delta\mathbf{Z} = \mathbf{X}(\widetilde{\mathbf{W}} - \mathbf{W})^\top$ and $\mathbf{H}_\mathbf{Z}$ denotes the Hessian of the loss with respect to the layer output. Since $\mathbf{Z} \in \mathbb{R}^{N\times O}$, explicitly forming $\mathbf{H}_\mathbf{Z}$ requires $\mathcal{O}(N^2O^2)$ storage, making such an approach impractical for LLMs.

Lemma 3 (Residual form of truncated SVD). *Let $\mathbf{A} \in \mathbb{R}^{O\times I}$ admit the singular value decomposition*

$$\mathbf{A} = \sum_{i=1}^{q} \sigma_i\mathbf{u}_i\mathbf{v}_i^\top, \qquad q = \min(O, I), \tag{23}$$

where σ_i denotes the i-th singular value, and $\mathbf{u}_i$ and $\mathbf{v}_i$ are the corresponding left and right singular vectors. For any index set $\mathcal{R} \subseteq \{1, \dots, q\}$, define the truncated reconstruction

$$\mathrm{Trunc}(\mathbf{A}) = \sum_{i\in\mathcal{R}} \sigma_i\mathbf{u}_i\mathbf{v}_i^\top. \tag{24}$$

Then, the residual can be decomposed as

$$\mathbf{A} - \mathrm{Trunc}(\mathbf{A}) = \sum_{i \notin \mathcal{R}} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top. \tag{25}$$

Proof. By the SVD decomposition of $\mathbf{A}$, we have

$$\mathbf{A} = \sum_{i=1}^{q} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top. \tag{26}$$

Subtracting the truncated reconstruction gives

$$\mathbf{A} - \mathrm{Trunc}(\mathbf{A}) = \sum_{i=1}^{q} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top - \sum_{i \in \mathcal{R}} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \tag{27}$$

$$= \sum_{i \notin \mathcal{R}} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top. \tag{28}$$

This completes the proof.

A.2 Simultaneous Compression of Multiple Layers

The analysis in Sec. 3.2 considers the perturbation induced by compressing a single weight matrix. In practice, multiple linear layers are compressed simultaneously. We show that the proposed layer-wise objective naturally extends to this setting under the same first-order approximation.

Consider a model with L layers, all of which are compressed. For each layer $l \in \{0, \dots, L-1\}$, we define the corresponding loss gradient with respect to $\mathbf{W}_l$ as $\mathbf{G}_{\mathbf{W}l} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}_l}$.

A joint first-order Taylor expansion with respect to all compressed weight matrices gives

$$\Delta \mathcal{L}^2 \approx \left[\sum_{l=0}^{L-1} \left\langle \mathbf{W}_l - \widetilde{\mathbf{W}}_l, \mathbf{G}_{\mathbf{W}l} \right\rangle_F \right]^2 \tag{29}$$

$$\leq L \sum_{l=0}^{L-1} \left\langle \mathbf{W}_l - \widetilde{\mathbf{W}}_l, \mathbf{G}_{\mathbf{W}l} \right\rangle_F^2 \tag{30}$$

$$\leq L \sum_{l=0}^{L-1} O_l \left\| (\mathbf{W}_l - \widetilde{\mathbf{W}}_l) \mathbf{G}_{\mathbf{W}l}^\top \right\|_F^2, \tag{31}$$

where both inequalities follow from Cauchy-Schwarz, with the second applying the layer-wise bound in Eq. 8, and O_l denoting the output dimension of layer l.

Thus, up to layer-dependent constant factors O_l, the first-order upper bound decomposes into a sum of the same gradient-aware objectives introduced in Eq. 9. Consequently, for fixed per-layer ranks, each weight matrix can be compressed independently using the LTG transformation derived in Sec. 3.2. Cross-layer interactions arising because the perturbation of an earlier layer changes the input to subsequent layers are contained in the higher-order remainder of the Taylor expansion and are therefore not explicitly modeled by our first-order approximation.

B Implementation Details

Numerical Stability. In practice, the empirical gradient covariance $\mathbf{G}_{\mathbf{W}}^\top \mathbf{G}_{\mathbf{W}}$ may be positive semi-definite or become numerically ill-conditioned, which can make the whitening transformation unstable. We therefore apply a small diagonal damping term before factorization:

$$\mathbf{C}_{\mathbf{G}} = \mathbf{G}_{\mathbf{W}}^\top \mathbf{G}_{\mathbf{W}} + \epsilon \mathbf{I}, \tag{32}$$

where $\epsilon > 0$ is a small constant added to ensure that $\mathbf{C}_{\mathbf{G}}$ is positive definite. This guarantees that $\mathbf{G}$ is non-singular and therefore invertible, which is required for the subsequent derivation of Eq. 13.

Table 2: Effect of different whitening mechanisms used to construct $\mathbf{G} \in \mathbb{R}^{I\times I}$ for compressing OPT-350M.

WHITENING MECHANISM	PPL	COMPLEXITY
LDL	53.03663	$\sim \frac{1}{3}I^3$
Eigh	53.04274	$\mathcal{O}(I^3)$
SVD	53.03490	$\mathcal{O}(I^3)$
Cholesky	53.03591	$\sim \frac{1}{3}I^3$

Gradient Collection. For clarity, the derivation in Sec. 3 omits the batch dimension and directly uses the weight gradient $\mathbf{G}_\mathbf{W} = \partial\mathcal{L}/\partial\mathbf{W}$. In practical implementations, however, a linear layer receives batched activations $\mathbf{X} \in \mathbb{R}^{B\times N\times I}$, where B denotes the batch size. Let

$$\mathbf{G}_\mathbf{Y} = \frac{\partial\mathcal{L}}{\partial(\mathbf{X}\mathbf{W}^\top)} \in \mathbb{R}^{B\times N\times O} \tag{33}$$

denote the gradient with respect to the layer output. By default, PyTorch accumulates the weight gradient over the batch dimension, yielding a single matrix

$$\mathbf{G}_\mathbf{W} = \sum_{b=1}^{B} \mathbf{G}_{\mathbf{Y}_b}^\top \mathbf{X}_b, \tag{34}$$

thereby removing the batch dimension before the gradient statistics used for whitening are constructed.

To preserve the individual calibration samples when estimating the gradient covariance, we instead register forward and backward hooks to collect the input activation $\mathbf{X}_b$ and output gradient $\mathbf{G}_{\mathbf{Y}b}$ of each linear layer. We then explicitly construct the corresponding per-sample weight gradients as

$$\mathbf{G}_{\mathbf{W}b} = \mathbf{G}_{\mathbf{Y}_b}^\top \mathbf{X}_b \in \mathbb{R}^{O\times I}, \qquad b = 1, \ldots, B, \tag{35}$$

and accumulate their covariance according to

$$\mathbf{C}_\mathbf{G} = \sum_{b=1}^{B} \mathbf{G}_{\mathbf{W}_b}^\top \mathbf{G}_{\mathbf{W}b} + \epsilon\mathbf{I}. \tag{36}$$

This avoids mixing different calibration samples through the default PyTorch backward graph while remaining consistent with the gradient-aware objective derived in Sec. 3.2.

Effect of Different Whitening Mechanisms. Cholesky decomposition is not the only factorization that can be used to construct the whitening matrix $\mathbf{G}$ in Eq. (10). Since $\mathbf{G}_\mathbf{W}^\top\mathbf{G}_\mathbf{W}$ is a symmetric matrix, several factorization methods can produce an equivalent transformation satisfying the required whitening relation. We consider three alternatives to Cholesky: LDL decomposition, eigendecomposition, and SVD.

For LDL decomposition,

$$\mathbf{C}_\mathbf{G} = \mathbf{L}\mathbf{D}\mathbf{L}^\top, \qquad \mathbf{G} = \mathbf{L}\mathbf{D}^{1/2}. \tag{37}$$

For eigendecomposition,

$$\mathbf{C}_\mathbf{G} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top, \qquad \mathbf{G} = \mathbf{Q}\mathbf{\Lambda}^{1/2}. \tag{38}$$

Similarly, using SVD,

$$\mathbf{C}_\mathbf{G} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \qquad \mathbf{G} = \mathbf{U}\mathbf{\Sigma}^{1/2}, \tag{39}$$

where for the symmetric positive-definite matrix considered here, $\mathbf{U} = \mathbf{V}$ up to numerical precision. Table 2 compares these whitening mechanisms when applied to `LTG` for compressing OPT-350M at a 40% compression ratio. All four choices yield nearly identical perplexity, indicating that the effectiveness of `LTG` is largely insensitive to the particular factorization used to construct $\mathbf{G}$. We therefore adopt Cholesky factorization throughout our experiments because it avoids explicitly computing singular or eigenvectors and requires lower computational cost.

Table 3: Performance of OPT-350M compressed by different methods across varying compression ratios, before and after reconstruction. C4 is evaluated using perplexity (PPL↓), while the six downstream datasets are evaluated using individual and average accuracy (↑). Results before and after reconstruction are shown with red and green backgrounds. Within each setting, the best result is highlighted in bold and the second-best result is underlined.

RATIO	METHOD	**PPL↓**	OPENB.↑	ARC_E↑	WINOG.↑	HELLAS.↑	PIQA↑	MATHQA↑	**AVERAGE↑**
0%	Original	21.15	0.282	0.404	0.523	0.367	0.647	0.226	0.408
20%	SVD-LLM	34.99	0.270	0.366	0.514	0.320	0.588	0.215	0.379
	LTG†	31.14	**0.284**	**0.382**	**0.515**	**0.332**	0.601	0.217	0.388
	LTG	**28.67**	0.278	**0.382**	**0.515**	0.327	**0.619**	**0.223**	**0.391**
40%	SVD-LLM	94.48	**0.278**	0.305	0.490	0.287	0.542	0.203	0.351
	LTG†	65.37	0.274	0.357	**0.511**	**0.302**	0.575	0.214	0.372
	LTG	**54.83**	**0.278**	**0.362**	0.508	0.300	**0.587**	**0.218**	**0.375**
60%	SVD-LLM	693.9	0.238	0.287	0.479	0.275	0.530	0.197	0.334
	LTG†	303.8	**0.256**	0.303	0.493	**0.276**	0.527	**0.199**	0.342
	LTG	**285.7**	0.254	**0.307**	**0.510**	0.270	**0.538**	**0.199**	**0.346**
80%	SVD-LLM	3291	0.250	0.269	0.495	0.257	**0.507**	0.200	0.330
	LTG†	1791	**0.270**	0.263	0.493	0.257	0.499	**0.205**	0.331
	LTG	**1608**	0.268	**0.282**	**0.499**	**0.264**	0.505	0.199	**0.336**
20%	SVD-LLM	34.35	0.284	0.375	0.507	0.331	0.619	0.220	0.389
	LTG†	30.19	0.270	**0.393**	0.502	0.329	**0.627**	**0.231**	0.392
	LTG	**27.90**	**0.288**	0.383	**0.514**	**0.333**	0.621	0.228	**0.395**
40%	SVD-LLM	63.12	0.274	0.347	0.492	0.298	0.576	0.209	0.366
	LTG†	44.45	0.264	**0.367**	**0.532**	**0.305**	0.592	0.215	0.379
	LTG	**36.44**	**0.286**	0.363	0.525	0.302	**0.593**	**0.221**	**0.382**
60%	SVD-LLM	285.3	0.244	0.304	0.508	0.275	0.554	**0.209**	0.349
	LTG†	96.73	0.246	0.329	0.506	**0.281**	0.566	0.207	0.356
	LTG	**61.74**	**0.248**	**0.336**	**0.526**	0.274	**0.571**	0.207	**0.360**
80%	SVD-LLM	822.7	0.250	0.292	0.504	0.260	0.528	0.204	0.340
	LTG†	430.6	0.252	0.298	**0.508**	**0.264**	0.529	0.197	0.341
	LTG	**186.8**	**0.268**	**0.314**	0.500	0.256	**0.530**	**0.206**	**0.346**

Table 4: Performance of OPT-350M compressed by different methods across varying compression ratios. C4 is evaluated using perplexity (PPL↓), while the six downstream datasets are evaluated using individual and average accuracy (↑).

RATIO	METHOD	PPL↓	OPENB.↑	ARC_E↑	WINOG.↑	HELLAS.↑	PIQA↑	MATHQA↑	**AVERAGE↑**
0%	Original	21.15	0.282	0.404	0.523	0.367	0.647	0.226	0.408
20%	SVD	141.0	0.260	0.315	0.493	0.285	0.552	0.212	0.353
	FWSVD	100.7	0.244	0.337	0.497	0.287	0.554	0.218	0.356
	ASVD	56.45	0.262	0.344	0.507	0.308	0.571	0.223	0.369
40%	SVD	467.5	0.230	0.301	0.517	0.266	0.526	0.210	0.342
	FWSVD	687.8	0.250	0.294	0.494	0.261	0.523	0.199	0.337
	ASVD	259.1	0.244	0.288	0.491	0.278	0.533	0.213	0.341
60%	SVD	1782	0.238	0.262	0.508	0.264	0.514	0.198	0.331
	FWSVD	2997	0.256	0.279	0.519	0.258	0.510	0.210	0.339
	ASVD	1779	0.234	0.265	0.503	0.260	0.510	0.203	0.329
80%	SVD	2241	0.246	0.272	0.512	0.258	0.503	0.203	0.332
	FWSVD	4160	0.252	0.269	0.511	0.257	0.509	0.209	0.334
	ASVD	2469	0.256	0.276	0.507	0.258	0.504	0.202	0.334

C ADDITIONAL EXPERIMENTAL RESULTS

Additional Baseline Comparison on OPT-350M. Table 3 provides the detailed numerical results corresponding to Fig. 4, while Table 4 complements these results by extending the comparison to additional representative SVD-based baselines, including SVD, ASVD, and FWSVD.

Compression Performance without Reconstruction. Table 5 presents the results of compressing different LLMs using SVD-LLM, LTG, and LTG† without reconstruction. Overall, both LTG and

Table 5: Performance of different LLMs across varying compression ratios using SVD-LLM, LTG†, and LTG without reconstruction. We report C4 perplexity (PPL↓), average downstream accuracy (Acc.↑), and GSM8K exact-match accuracy (GSM8K↑). The best and second-best results under each compression ratio are highlighted in bold and underlined, respectively.

Ratio	Method	OPT-6.7B			Vicuna-7B			LLaMA 2-7B			LLaMA-7B			Mistral-7B		
		PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑	PPL↓	Acc.↑	GSM8K↑
20%	SVD-LLM	15.98	0.536	0.014	21.34	0.497	0.029	18.46	0.484	0.017	14.95	0.524	0.024	22.76	0.539	0.019
	LTG†	15.44	0.537	**0.017**	19.48	**0.542**	**0.044**	17.98	**0.539**	0.029	14.77	**0.543**	**0.039**	21.96	**0.546**	0.030
	LTG	**14.27**	**0.542**	**0.017**	**15.55**	0.524	0.022	**13.09**	0.523	**0.031**	**11.69**	0.534	0.025	**14.45**	0.541	**0.043**
40%	SVD-LLM	31.97	0.433	0.008	70.08	0.408	0.011	81.72	0.398	0.015	50.53	0.422	0.015	130.9	**0.409**	0.007
	LTG†	29.83	0.496	0.016	61.56	**0.444**	**0.023**	65.21	**0.425**	0.016	49.40	0.439	**0.024**	85.36	0.389	0.008
	LTG	**20.25**	**0.504**	**0.017**	**42.67**	0.423	0.014	**38.21**	0.415	**0.017**	**26.89**	**0.441**	0.018	**44.44**	0.398	**0.014**
60%	SVD-LLM	374.0	0.335	0.001	673.7	0.337	0.006	529.0	0.332	0.009	359.3	0.349	0.004	739.4	0.337	**0.002**
	LTG†	338.6	0.389	0.006	**379.1**	0.341	**0.008**	486.2	0.336	**0.011**	331.0	0.343	**0.007**	561.5	0.338	**0.002**
	LTG	**80.51**	**0.408**	**0.008**	428.6	**0.354**	0.005	**220.6**	**0.344**	0.002	**126.8**	**0.352**	0.006	**208.4**	**0.341**	**0.002**
80%	SVD-LLM	12696	0.334	**0.000**	2649	0.327	0.000	3146	0.326	**0.002**	2657	0.330	0.002	4591	0.330	0.000
	LTG†	**7716**	0.332	**0.000**	**2152**	0.327	**0.008**	1713	**0.331**	0.000	1372	0.324	**0.009**	2349	0.328	**0.004**
	LTG	11986	**0.335**	**0.000**	85156	**0.329**	0.000	**1181**	0.328	0.000	**756.3**	**0.332**	0.004	**1280**	**0.331**	0.000

Table 6: Squared loss pertubation $\Delta\mathcal{L}^2$ relative to the original OPT-350M model after removing each singular component chunk using SVD, SVD-LLM, and LTG†, averaged over three random seeds (3, 42, and 100). This experiment extends Fig. 3a to multiple seeds to assess the robustness of the observed loss perturbation patterns. Lower values indicate better preservation of the original model objective.

Method	Chunk 0	Chunk 1	Chunk 2	Chunk 3	Chunk 4
SVD	2.82E+02 ± 0E+00	5.22E+00 ± 0E+00	8.20E-01 ± 0E+00	2.60E-01 ± 0E+00	9.00E-02 ± 0E+00
SVD-LLM	7.89E+01 ± 6.57E+00	4.38E+00 ± 3.00E-02	7.50E-01 ± 9.68E-03	1.90E-01 ± 1.37E-03	7.00E-02 ± 1.42E-03
LTG†	8.88E+01 ± 6.24E+00	4.86E+00 ± 9.00E-02	8.20E-01 ± 2.00E-02	1.90E-01 ± 2.75E-03	6.00E-02 ± 1.61E-04
Method	**Chunk 5**	**Chunk 6**	**Chunk 7**	**Chunk 8**	**Chunk 9**
SVD	4.00E-02 ± 0E+00	2.00E-02 ± 0E+00	1.00E-02 ± 0E+00	8.80E-03 ± 0E+00	1.00E-01 ± 0E+00
SVD-LLM	2.00E-02 ± 2.41E-04	9.28E-03 ± 3.86E-05	4.32E-03 ± 1.48E-04	2.05E-03 ± 3.21E-05	1.88E-03 ± 5.16E-05
LTG†	2.00E-02 ± 9.55E-05	7.68E-03 ± 6.93E-05	3.26E-03 ± 3.65E-05	1.37E-03 ± 2.47E-05	5.14E-04 ± 3.15E-06

LTG† outperform SVD-LLM across most settings, with particularly substantial gains in perplexity. For instance, at 60% compression rate, LTG reduces the perplexity of OPT-6.7B from 374.0 to 80.51, corresponding to a $4.6\times$ reduction relative to SVD-LLM.

Vicuna-7B exhibits a distinct trend compared with the other evaluated models, with the externally calibrated LTG† outperforming the self-calibrated LTG. This may reflect the instruction-tuned nature of Vicuna-7B, which is derived from LLaMA 2 and further adapted using instruction-following data. As a result, self-generated calibration sequences may not fully represent the broader range of contexts encountered during instruction tuning, making external calibration data comparatively more informative.

At the most aggressive compression ratio of 80%, LTG exhibits substantially higher perplexity on OPT-6.7B and Vicuna-7B than the competing baselines. One possible explanation is that the proposed compression objective is derived from the first-order Taylor expansion, whose local approximation may become less accurate under the large perturbations induced by extreme compression. Importantly, as shown in Table 1, this degradation is largely mitigated after the reconstruction phase. Moreover, we do not observe the same behavior in the other evaluated model and compression settings.

Robustness Across Random Seeds. To assess the robustness of the loss-perturbation analysis in Fig. 3a, we repeat the experiment using three random seeds (3, 42, and 100). For each method, we measure the squared loss change $\Delta\mathcal{L}^2$ relative to the original OPT-350M model after removing each singular-component chunk. Since the random seed only affects calibration-data sampling and the estimation of the gradient-based whitening transformation, vanilla SVD remains deterministic and therefore yields zero standard deviation across runs. As shown in Table 6, the overall trend remains consistent across seeds, confirming our observations.

Table 7: Model memory footprint across different compression ratios, reported in GiB.

RATIO	OPT-350M	OPT-6.7B	VICUNA-7B	LLAMA-2-7B	LLAMA-7B	MISTRAL-7B
0%	1.24	12.40	12.55	12.55	12.55	13.49
20%	1.01	10.00	10.20	10.20	10.20	11.89
40%	0.78	7.60	7.79	7.79	7.79	9.29
60%	0.56	5.20	5.38	5.38	5.38	6.69
80%	0.33	2.80	2.96	2.96	2.96	4.09

Table 8: Comparison of LTG with structured pruning methods under matched memory budgets for compressing LLaMA-7B. All methods use WikiText-2 as the external calibration dataset, whereas LTG relies on self-generated calibration data. We report the resulting perplexity on C4, with the best result under each memory budget highlighted in bold.

METHOD	13.5 GB	10 GB	9 GB	8 GB	7 GB
Original	7.34	–	–	–	–
LLM-Pruner	–	14.40	21.60	37.73	63.55
BlockPruner	–	15.87	22.99	43.42	88.55
SliceGPT	–	59.22	101.7	161.8	258.4
LTG†	–	14.19	16.82	20.97	27.53
LTG	–	**11.14**	**12.16**	**13.84**	**16.34**

Detailed Compressed Size of Each Models. Table 7 reports the storage requirements of the compressed checkpoints across different compression ratios. As all baselines use the same uniform compression ratio across layers, they produce the same model size under each compression setting.

Comparison with Structured Pruning Methods. We further evaluate LTG against state-of-the-art structured pruning baselines, including LLM-Pruner (Ma et al., 2023), BlockPruner (Zhong et al., 2025), and SliceGPT (Ashkboos et al., 2024), following the experimental setup described in Sec. 4.1. Table 8 reports the comparison under matched memory budgets on LLaMA-7B. LTG consistently achieves lower C4 perplexity across all evaluated budgets while relying only on self-generated calibration data, whereas the competing methods use an external calibration corpus. The performance gap becomes larger as the memory budget decreases, indicating that LTG better preserves model quality under more aggressive compression.

D LIMITATIONS

A limitation of LTG is its higher memory requirement during the compression process compared with activation-based approaches. In addition to the model parameters and activations, LTG requires storing gradient information to estimate the local loss sensitivity, which can sustainably increase the memory usage of the compression stage in our implementation.

This overhead can be reduced through engineering implementation such as gradient checkpointing, which trades additional computation for lower memory usage. However, this would increase the overall compression time. We consider this trade-off acceptable in the intended deployment setting, where compression is performed offline on a sufficiently powerful server and the resulting compressed model is subsequently deployed to resource-constrained end devices. Under this setting, the additional memory and computation are incurred only once during compression and do not affect inference-time efficiency on the target device.

In practice, the memory requirement remains manageable for the model scales considered in this work. All experiments reported in this paper are conducted on a single NVIDIA L40S GPU with 48 GB of VRAM.