SINGULAR ENERGY-AWARE LAYER-WISE RANK ALLOCATION FOR ACCURATE LOW-RANK FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning large-scale neural networks in a low-rank space has become a popular strategy in the era of large language models (LLMs). However, most existing methods, such as low-rank reparameterization and low-rank adaptation (LoRA), assume a uniform rank across all layers. We provide practical insights into exploiting the learning capacity of low-rank models through layer-wise rank allocation. In particular, we find that the singular energy of gradients plays a key role in determining layer-wise ranks for strong downstream performance. Based on this observation, we propose a singular energy-aware (SEA) rank allocation scheme that can be applied to a broad class of low-rank fine-tuning methods. We further provide a theoretical analysis of the optimal rank in terms of retained energy after pruning. Extensive experiments demonstrate consistent improvements in downstream performance with negligible additional computational and memory overhead.

1 INTRODUCTION

Large-scale neural networks have been widely adopted across a myriad of applications, achieving unprecedented success. In particular, attention-based networks play a key role in delivering outstanding performance. However, deploying such models in real-world applications raises significant concerns regarding efficient fine-tuning and inference, due to their ever-increasing number of parameters. Several techniques have been proposed to address this issue, including structured pruning, low-rank adaptation (LoRA) (Hu et al., 2022), and their variants.

Most existing low-rank reparameterization methods as well as LoRA variants adopt uniform rank allocation, assigning the same rank to all layers. While this simplifies implementation, our study reveals that it increases the optimization error, i.e., the discrepancy between the full-rank and low-rank updates, which in turn degrades model performance. Although some existing works consider layer-wise characteristics, such as AdaLoRA (Zhang et al., 2023), SVD-LLMv2 (Wang et al., 2025b), and WeLore (Jaiswal et al., 2025), they either incur non-negligible computational overhead for adaptive rank allocation or yield marginal improvements over uniform rank allocation. In this study, we address the following fundamental question: *how should the rank be allocated across layers?*

In this paper, we formulate the layer-wise rank allocation problem as a convex optimization problem and propose a practical solution to determine layer-wise rank assignments for efficient and accurate low-rank fine-tuning. First, we quantify the importance of individual singular modes based on their relative energy within each layer. We then formulate a maximization problem to maximize the retained energy after structured pruning under a predefined rank budget. This formulation enables efficient and principled identification of the optimal layer-wise ranks. Our study further reveals that, when fine-tuning a pre-trained model on a downstream dataset, the importance of singular modes should be measured from gradients rather than weight parameters. Based on this observation, we adopt a simple and efficient gradient calibration method. Putting all together, we introduce a general Singular Energy-Aware (SEA) rank allocation scheme that can be readily applied to low-rank reparameterization and LoRA-based methods.

To evaluate our proposed rank allocation scheme, we conduct experiments on several representative benchmarks using popular large language models (LLMs), including OPT, Phi, Qwen, and LLaMA. We show that the SEA rank allocation scheme identifies optimal layer-wise ranks for efficient and accurate fine-tuning. For example, when fine-tuning OPT-350M with low-rank reparameterization on GLUE, SEA achieves up to a 7% improvement in average validation performance over the uniform

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.

Table 1: Fundamental conceptual differences among representative rank allocation schemes.

Features	**Adaptive Rank Allocation in AdaLoRA**	**Static Rank Allocation in SVD-LLMv2**	**Singular Energy-Aware Rank Allocation (proposed)**
Importance Score	The product of sensitivity $\lvert w \cdot \nabla_w \ell \rvert$	Activation-based truncation loss	Locally normalized gradient singular energy
Importance Score Implication	Loss change	Layer's output change	Singular energy change in gradients
Budget Allocation	Global budget allocation	Group-wise budget allocation	Global budget allocation
Layer-scale bias	Not normalized $\rightarrow$ possible bias	Group-wise log-normalization $\rightarrow$ bias alleviated	Layer-wise normalization $\rightarrow$ bias avoided
Scheduling	Dynamic	Static	Static
Extra Computations	Inner-product of weights and gradients	Truncation-loss calibration (once)	Gradient calibration (once)

rank scheme. SEA also outperforms state-of-the-art (SOTA) LoRA variants that adapt layer-wise ranks, including AdaLoRA and FlexLoRA. These results suggest that existing low-rank training methods can benefit from SEA to better utilize model capacity under the same parameter budget.

2 Related work

We first contrast SEA with two representative rank allocation schemes, the dynamic allocation of AdaLoRA (Zhang et al., 2023) and the static allocation of SVD-LLMv2 (Wang et al., 2025b) and DobiSVD (Wang et al., 2025a). SEA differs in two respects: it uses the change in gradient singular energy as the importance signal, and it normalizes energy within each layer, avoiding layer-scale bias. This comparative analysis shows which aspects our study focuses on and why SEA achieves higher accuracy than existing rank allocation methods.

Layer-wise rank allocation methods for low-rank pruning – EigenDamage (Wang et al., 2019) jointly leverages Fisher information and singular value energy, but the Fisher information is expensive to compute and memory-intensive. A layer-wise independent rank selection method (Guo & Yu, 2025) ignores inter-layer interactions, yielding suboptimal overall performance. SVD-NAS (Yu & Bouganis, 2023) casts rank allocation as a neural architecture search problem, offering flexible trade-offs at a high computational cost. MoDeGPT (Lin et al., 2025) performs module-wise decomposition to achieve structured compression by reducing hidden dimensionality. WeLore (Jaiswal et al., 2025) instead allocates rank from a min-max normalized threshold search over the pretrained weight spectrum alone, requiring no downstream-data signal.

Layer-wise rank allocation methods for low-rank adapters – Most LoRA variants assume a uniform rank across layers, and only a few studies have explored layer-wise rank allocation. As discussed above, AdaLoRA (Zhang et al., 2023) adaptively determines layer-wise ranks during fine-tuning based on a sensitivity score. FlexLoRA (Liu et al., 2026) instead allocates ranks across adapters based on a spectral-entropy score to balance parameter efficiency and performance, and IGU-LoRA (Cui et al., 2026) allocates layer-wise ranks using integrated gradients. However, these importance metrics are computed iteratively during training, which introduces extra overhead and can be susceptible to gradient noise across steps, potentially destabilizing training.

3 Method

3.1 Motivational empirical study

Gradients lie in a lower-rank subspace than weights. Figure 2 presents the layer-wise rank required to capture 90% of the singular value energy, where brighter colors indicate a higher required rank.

The left heatmap shows ranks measured from LLaMA-3.2-3B weights, while the right heatmap shows ranks measured from gradients during fine-tuning on the Commonsense dataset. Across all layers, the

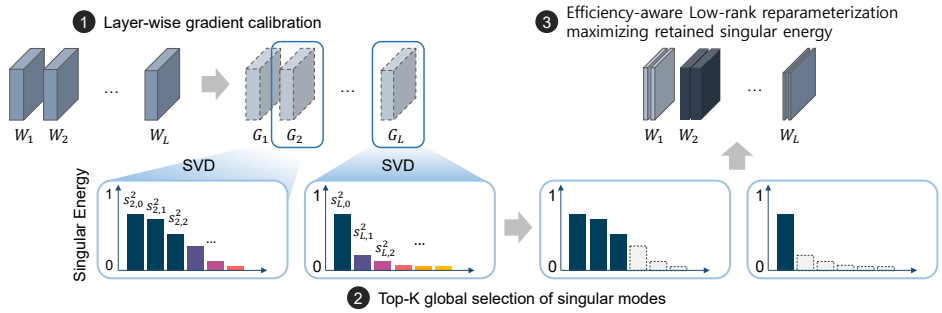

Figure 1: Schematic illustration of proposed singular energy-aware (SEA) rank allocation method. W_l and G_l are weights and gradients at layer l, respectively. SEA allocates greater ranks to the layers with a flatter spectrum relative to its parameter cost.

weight tensors appear significantly brighter than the gradient tensors, and this trend holds consistently across different benchmarks (See Appendix for additional results), indicating that updates occur in a much lower-rank subspace than the weights themselves.

Gradients as a better metric than weights. Fine-tuning aims to adapt a model to downstream tasks whose data distributions often differ from those seen during pre-training, so the singular spectrum of the resulting gradients can be substantially different from that of the pre-trained weights. For instance, layers 8 and 34 appear in darker red on the left (weight) map, indicating relatively lower required ranks, but appear substantially brighter on the right (gradient) map, indicating much higher required ranks. This misalignment implies that weight-based allocation may leave task-critical layers under-parameterized, making gradients a more appropriate proxy for identifying important singular modes for rank allocation, and motivating our focus on the gradient spectrum.

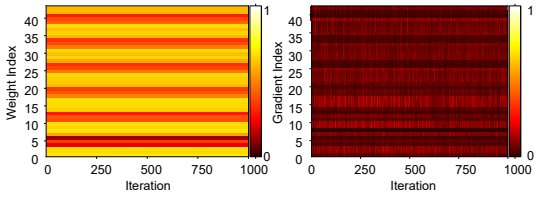

Figure 2: Normalized layer-wise ranks capturing 90% of the singular energy across 42 weight tensors (left) and their gradients (right) of LLaMA 3.2-3B during Commonsense fine-tuning.

Low-rank reparameterization can be applied statically. Another key observation is that the singular spectra of both weights and gradients remain largely consistent across fine-tuning steps. Since the learning rate is typically small (e.g., 1e-4 or 1e-5), the shape of the spectra changes only marginally. Although the gradient spectrum is relatively less stable than that of the weights, its fluctuations are not large enough to alter the appropriate rank. Therefore, we find that a static rank allocation scheme is well-suited for low-rank fine-tuning.

3.2 Suboptimality of Uniform Rank Allocation: A Theoretical Perspective

Consider a network of L layers, and let $M_l \in \mathbb{R}^{d_l^{out} \times d_l^{in}}$ denote the matrix associated with layer l, whose singular spectrum is used to allocate the rank of that layer. The analysis in this section holds for any such choice, and we will specify the matrix in the next section.

Let $M_l = U_l \Sigma_l V_l^\top$ be its singular value decomposition, where $\Sigma_l = \mathrm{diag}(\sigma_{l,1}, \ldots, \sigma_{l,d_l})$ with $d_l = \min(d_l^{in}, d_l^{out})$ and $\sigma_{l,1} \geq \sigma_{l,2} \geq \cdots \geq \sigma_{l,d_l} \geq 0$.

The energy carried by the i-th singular mode is $\sigma_{l,i}^2$. We measure it relative to the total energy,

$$s_{l,i} := \frac{\sigma_{l,i}^2}{\sum_{j=1}^{d_l} \sigma_{l,j}^2}, \tag{1}$$

so that modes from layers with different spectral scales are comparable. Truncating layer l to rank r_l discards

$$E_l(r_l) := \sum_{i>r_l} s_{l,i}, \tag{2}$$

which we call *tail energy* of layer l. Let c_l denote the number of parameters required per unit of rank at layer l. Then, a rank allocation $\mathbf{r} = (r_1, \ldots, r_L)$ requires $\sum_l c_l r_l$ parameters. Our goal is to choose $\mathbf{r}$ to minimize $\sum_l E_l(r_l)$ subject to the parameter budget P_{budget}.

Then, we present our theoretical results as follows. See Appendix for proofs.

Lemma 1 (Convexity of the tail energy)**.** *For every layer l, the normalized tail energy E_l defined in (2) is convex in r_l. Its piecewise-linear interpolation is convex as well.*

Theorem 2. *The global optimal rank allocation $\mathbf{r}^*$ across L layers that minimizes the normalized tail energy $\sum_l E_l(r_l)$ under a fixed parameter budget $\sum_l c_l r_l = P_{budget}$ satisfies*

$$\exists\, \lambda \geq 0 \quad \textit{s.t.} \quad \frac{s_{l,\, r_l^*+1}}{c_l} \leq \lambda \leq \frac{s_{l,\, r_l^*}}{c_l} \quad \textit{for all } l \in [L]. \tag{3}$$

Here, r_l is relaxed to be continuous. At a layer whose optimal rank is not integral, the two bounds in (3) coincide.

Remark 1 (Interpretation of the threshold λ)**.** *Theorem 2 shows that the optimal allocation is governed by a single threshold λ shared by all layers. The ratio $s_{l,k}/c_l$ can be viewed as the energy density of the k-th singular component in layer l, i.e., the normalized energy it retains per parameter. Since $s_{l,k}$ is non-increasing in k (Lemma 1), the optimal rank r_l^* keeps exactly the components whose energy density is above λ and drops those below it. Consequently, once λ is fixed, each layer can determine its rank independently of the other layers.*

Corollary 3 (Suboptimality of uniform rank allocation)**.** *Let $r^{uni} = \lfloor P_{budget} / \sum_l c_l \rfloor$ denote the uniform rank. Then, $\sum_l E_l(r^{uni}) \geq \sum_l E_l(r_l^*)$, and the inequality is strict unless* (3) *holds at r^{uni}, that is, unless some single λ satisfies $s_{l,r^{uni}+1}/c_l \leq \lambda \leq s_{l,r^{uni}}/c_l$ in every layer.*

Remark 2 (Method-agnostic scope)**.** *Theorem 2 and Corollary 3 do not depend on how the rank r_l is parameterized. The analysis involves only the singular value spectra of the gradient matrices and the per-rank parameter costs c_l. Thus, both results hold for any low-rank scheme whose per-layer cost is $c_l r_l$, including low-rank weight reparameterization and LoRA-type adaptation.*

Remark 3 (Model-agnostic scope)**.** *Our analysis represents each layer as a matrix and makes no further assumption on the network architecture. The results therefore apply to a broad range of architectures. In particular, uniform rank allocation is strictly suboptimal unless a single λ satisfies (3) at $\mathbf{r}^{uni}$, which rarely happens when the energy densities $s_{l,k}/c_l$ differ across layers.*

3.3 Singular Energy-Aware (SEA) rank allocation

The preceding analysis characterizes the optimal allocation for any choice of M_l. There are two remaining questions: 1) which matrix should determine the spectrum, and 2) how the threshold λ should be found under an integer budget. We answer the first by using the gradient matrix rather than the pre-trained weight matrix, and the second with a greedy top-K procedure.

Gradient energy calibration – The pre-trained weight spectrum is fixed before fine-tuning begins and therefore yields the same allocation regardless of the downstream task. The gradient spectrum, in contrast, is determined by the downstream data. Under a first-order approximation, discarding the leading singular directions of the gradient causes the largest increase in the loss. Our answer to the first question is therefore that layer-wise ranks should be determined by the gradient rather than by the pre-trained weight.

Computing the full-batch gradient is computationally expensive. A straightforward solution is to approximate the importance score $s_{l,i}$ using stochastic gradients computed from a few mini-batches. We refer to this step as *gradient calibration*. Rather than using randomly sampled data, we perform gradient calibration using a small set of high-loss samples as follows:

$$\ell_1 \geq \ell_2 \geq \cdots \geq \ell_m$$

$$\hat{G} = \frac{1}{k} \sum_{i=1}^{k} \nabla \ell_i, \tag{4}$$

where ℓ_i denotes the loss of the i-th sample, m is the mini-batch size, and k is the number of calibration samples. In practice, when m is much smaller than the size of the downstream dataset, one can run multiple iterations and average the resulting $\hat{G}$ across iterations.

In general, commonly used loss functions, such as cross-entropy and mean squared error, exhibit gradient magnitudes that tend to correlate with the loss values (Katharopoulos & Fleuret, 2018). Based on this observation, we expect the following relation to hold:

$$\|G\|_F \approx \|\hat{G}\|_F \gg \|R\|_F := \left\| \frac{1}{m-k} \sum_{i=k+1}^{m} \nabla \ell_i \right\|_F ,$$

where R denotes the aggregated gradient over low-loss samples. This suggests that high-loss samples dominate gradient energy, enabling a more accurate approximation of full-batch singular modes than random samples. Thus, we set $M = \hat{G}$ in the remainder of this paper.

How to maximize the retained energy after pruning – To determine the layer-wise rank configuration that maximally preserves relative energy while minimizing the total parameters, we formulate an optimization problem as follows. Let P_{budget} be the target budget and $x_{l,i} \in \{0, 1\}$ be binary variables indicating whether each singular mode is preserved. Then, we solve the following problem:

$$\max_{x_{l,i}} \sum_{l,i} s_{l,i} x_{l,i} \quad \text{s.t.} \quad \sum_{l,i} c_l x_{l,i} \leq P_{\text{budget}}, \tag{5}$$

where c_l is the parameter cost per mode at layer l. Using a Lagrangian multiplier λ, we define:

$$\begin{aligned} \mathcal{L}(x, \lambda) &= \sum_{l,i} s_{l,i} x_{l,i} - \lambda \left(\sum_{l,i} c_l x_{l,i} - P_{\text{budget}} \right) \\ &= \sum_{l,i} (s_{l,i} - \lambda c_l) x_{l,i} + \lambda P_{\text{budget}}. \end{aligned} \tag{6}$$

For a fixed λ, maximizing $\mathcal{L}$ with respect to $x_{l,i}$ implies $x_{l,i} = 1$ if $s_{l,i} - \lambda c_l > 0$, and $x_{l,i} = 0$ otherwise. This yields the solution:

$$x^*_{l,i} = \mathbf{1}\left(\frac{s_{l,i}}{c_l} > \lambda^* \right), \tag{7}$$

where λ^* is the smallest multiplier for which the budget is not exceeded, that is,

$$\begin{aligned} C(\lambda) &:= \sum_{l,i} c_l \mathbf{1}\left(\frac{s_{l,i}}{c_l} > \lambda \right) \\ \lambda^* &:= \inf \left\{ \lambda \geq 0 : C(\lambda) \leq P_{\text{budget}} \right\}. \end{aligned} \tag{8}$$

Near-optimality of the threshold solution – Throughout this paper, x^* denotes the threshold solution given by (7), that is, $x^*_{l,i} = \mathbf{1}(s_{l,i}/c_l > \lambda^*)$, whose total parameter cost is $\sum_{l,i} c_l x^*_{l,i} = C(\lambda^*)$. By the definition of λ^* we have $C(\lambda^*) \leq P_{\text{budget}}$, so x^* is feasible for (5). However, since $C(\lambda)$ is a non-increasing step function, this inequality is likely strict, leaving a small portion of the budget, $\Delta := P_{\text{budget}} - C(\lambda^*)$, unused. The threshold solution x^* is therefore not guaranteed to be an exact maximizer of (5). The following result shows that the loss incurred by the unused budget is bounded by the energy of a single singular mode.

Proposition 4. *Let x^{opt} be an optimal solution of* (5) *and let $S(x) := \sum_{l,i} s_{l,i} x_{l,i}$. Assume that the densities $s_{l,i}/c_l$ are distinct and let (l_b, i_b) denote the discarded mode with the largest density. Then*

$$S(x^{opt}) - S(x^*) \leq \frac{\Delta}{c_{l_b}} s_{l_b,i_b} < s_{l_b,i_b}. \tag{9}$$

Remark 4 (Interpretation of near-optimality)**.** *Proposition 4 shows that the energy lost by the threshold solution $\mathbf{x}^*$ is smaller than the energy of a single discarded mode (l_b, i_b). In practice, this energy is small because gradients have been shown to be approximately low-rank (Sonthalia et al., 2025), which concentrates most of the layer energy in a few leading modes. Thus, the boundary mode is expected to account for only a small portion of the total retained energy $S(\mathbf{x}^{opt})$.*

Algorithm 1 Singular Energy-Aware (SEA) Rank Allocation

Require: Layer sizes $\{c_l\}$, gradients $\{G_l\}$, parameter budget P_{budget}.

Construct modes $\mathcal{M} = \{(l, i)\}$ from $\{G_l\}$ via SVD.
for each $(l, i) \in \mathcal{M}$ **do**
  Compute energy density $s_{l,i}/c_l$.
end for
Sort modes by $s_{l,i}/c_l$ in descending order.
$C \leftarrow 0$, $x_{l,i} \leftarrow 0$ for all (l, i).
for each (l, i) in sorted order **do**
  if $C + c_l > P_{budget}$ **then**
    break
  end if
  $x_{l,i} \leftarrow 1$; $C \leftarrow C + c_l$
end for
for each layer l **do**
  $r_l = \max(1, \sum_i x_{l,i})$
end for
return $\{r_l\}$.

Algorithm 2 SEA-based Reparameterization / LoRA

Require: Layer sizes $\{c_l\}$, pre-trained model $\mathbf{w}$, budget P_{budget}.

$\{G_l\} \leftarrow$ Gradient calibration (Eq. (4)).
$\{r_l\} \leftarrow$ Rank allocation($\{G_l\}$) (Alg. 1).
$\bar{\mathbf{w}} \leftarrow$ Low-rank reparam./adapter($\mathbf{w}, \{r_l\}$).
return $\bar{\mathbf{w}}$.

How to calculate layer-wise ranks via Top-K mode selection – Based on our analysis, now we can answer the second question: how the threshold λ should be found. We propose a Top-K selection method that still guarantees near-optimal retained singular energy. Algorithm 1 presents the Singular Energy-Aware (SEA) rank allocation method. Given the layer-wise singular modes obtained from a separate gradient calibration step, it calculates the density values $s_{l,i}/c_l$ for all individual layers and their singular modes. Then, it sorts the singular modes in decreasing order based on the density values and selects modes from the top of the sorted list until the budget P_{budget} is exhausted. *The density value of the last selected mode is exactly the threshold λ, which is thus determined implicitly without an explicit search.* Based on Lemma 1 and Theorem 2, this Top-K approach satisfies the bound in (9), that is the retained energy gap is bounded by the energy of the discarded mode with the largest density.

Summing (7) over the modes of layer l gives the rank assigned to that layer,

$$r_l \;=\; \sum_i x^*_{l,i} \;=\; \left|\left\{\, i \;:\; s_{l,i} > \lambda^* c_l \,\right\}\right|, \tag{10}$$

where $|\cdot|$ returns the number of elements. By (1), the normalized energies of each layer sum to one, so the right-hand side counts how many modes of layer l carry more than a $\lambda^* c_l$ share of that layer's own energy. A layer whose energy concentrates in a few leading modes has few such modes and receives a small rank. In contrast, a layer whose energy spreads over many modes has many modes above the threshold and receives a large rank. The allocation is therefore governed by the flatness of each layer's gradient energy spectrum, measured relative to its per-rank cost c_l, which is the behavior illustrated in Figure 1.

3.4 Application to low-rank fine-tuning

Algorithm 2 outlines the SEA-based reparameterization and LoRA framework. Given the parameter costs of all singular modes $\{c_l\}$, the gradient matrices $\{G_l\}$ computed using a calibration set from the downstream dataset, and a parameter budget P_{budget}, the algorithm first determines the layer-wise ranks $\{r_l\}$. These ranks can then be directly applied to low-rank reparameterization or low-rank adapters, maximizing parameter efficiency under the given budget. Note that Algorithm 1 runs SVD for all layer-wise gradients. We will empirically demonstrate that the computational cost is not significant in the next section by presenting timing breakdowns.

Table 2: [Low-rank reparameterization] GLUE (OPT-350M) and Commonsense (LLaMA 3.2-3B) fine-tuning performance comparisons across different rank allocation schemes. The models are compressed using the SVD-based reparameterization.

Method	Param.	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
Full FT	100%	84.43	90.36	90.00	93.65	81.64	84.72	83.09	68.14	84.50
Uniform Rank	75%	81.84	89.60	88.16	91.44	77.33	83.83	77.61	55.11	80.62
Weight-based		81.96	90.51	88.71	91.51	77.66	86.49	76.72	60.65	81.78
SEA (proposed)		**82.88**	**90.67**	**88.93**	**92.74**	**80.20**	**87.14**	**81.13**	**67.15**	**83.86**
Uniform Rank	50%	79.01	88.83	84.11	88.76	69.65	74.31	71.32	54.15	76.27
Weight-based		79.64	89.00	85.36	90.25	69.13	82.23	71.32	54.52	77.68
SEA (proposed)		**79.84**	**89.30**	**86.32**	**90.44**	**74.69**	**83.31**	**75.65**	**57.28**	**79.60**
Uniform Rank	25%	74.52	86.77	80.74	85.24	69.13	21.45	70.75	52.83	67.68
Weight-based		74.70	87.40	82.08	86.12	69.13	34.50	71.81	52.35	69.76
SEA (proposed)		**75.02**	**87.62**	**82.76**	**87.65**	**69.32**	**67.42**	**72.39**	**54.39**	**74.57**

Method	Param.	BoolQ	PiQA	HellaS.	Wino.	SIQA	ARC-e.	ARC-c.	OBQA	Avg
Full FT	100%	85.86	80.69	76.43	78.79	61.05	78.77	50.33	43.40	69.42
Uniform Rank	75%	82.44	75.73	55.06	67.40	**59.72**	68.07	38.13	**41.80**	61.04
Weight-based		84.34	75.19	54.24	**67.88**	56.60	64.39	37.46	38.60	59.84
SEA (proposed)		**85.50**	**76.61**	**58.60**	67.76	58.50	**68.42**	**43.48**	40.40	**62.41**
Uniform Rank	50%	67.65	68.01	34.82	56.43	51.13	47.19	30.43	31.20	48.36
Weight-based		64.95	67.68	32.47	55.74	49.39	45.96	27.09	30.80	46.76
SEA (proposed)		**76.94**	**70.24**	**38.32**	**57.59**	**52.87**	**52.28**	**33.78**	**33.60**	**51.95**
Uniform Rank	25%	63.24	62.95	26.92	49.96	44.58	36.32	23.41	24.60	41.50
Weight-based		63.64	63.76	27.32	50.96	43.76	36.49	21.40	23.60	41.37
SEA (proposed)		**64.86**	**64.45**	**27.56**	**51.37**	**45.90**	**38.95**	**23.91**	**27.70**	**43.09**

Table 3: [LoRA] Fine-tuning performance comparisons across different rank allocation schemes. **Top:** GLUE (OPT-350M). **Bottom:** Commonsense reasoning (LLaMA 3.2-3B). All methods are tuned to have the same number of parameters.

Method	Param.	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
Full FT	100%	84.43	90.36	90.00	93.65	81.64	84.72	83.09	68.14	84.50
Uniform Rank$_{r=32}$	3.66%	84.63	90.66	89.95	94.15	81.11	87.09	78.92	66.06	84.07
Weight-based		81.44	89.37	89.00	93.35	81.59	87.49	80.39	67.52	83.77
SEA (proposed)		**84.72**	**90.72**	**90.32**	**94.21**	**81.98**	**87.55**	**81.37**	**68.95**	**84.98**
Uniform Rank$_{r=8}$	0.94%	84.35	90.10	89.62	**94.38**	80.15	85.67	78.06	62.45	83.10
Weight-based		82.18	89.41	88.80	91.51	80.06	84.66	79.41	65.70	82.72
SEA (proposed)		**84.45**	**90.55**	**90.09**	94.27	**80.82**	**85.86**	**79.90**	**67.51**	**84.18**

Method	Param.	BoolQ	PiQA	HellaS.	Wino.	SIQA	ARC-e.	ARC-c.	OBQA	Avg
Full FT	100%	85.86	80.69	76.43	78.79	61.05	78.77	50.33	43.40	69.42
Uniform Rank$_{r=32}$	1.51%	87.09	80.52	75.38	76.80	62.64	77.54	52.17	46.80	69.87
Weight-based		87.28	81.34	76.09	79.41	62.03	80.00	52.84	50.00	71.12
SEA (proposed)		**88.62**	**81.66**	**77.60**	**80.66**	**63.46**	**81.23**	**54.18**	**52.80**	**72.53**
Uniform Rank$_{r=8}$	0.38%	87.19	80.52	75.04	77.45	61.67	74.56	53.18	44.80	69.30
Weight-based		86.33	**81.50**	74.59	78.49	61.67	77.19	52.17	45.80	69.72
SEA (proposed)		**87.22**	81.45	**77.08**	**79.91**	**63.41**	**79.65**	**55.18**	**51.60**	**71.94**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

We evaluate SEA on representative LLM benchmarks[1] – GLUE (Wang et al., 2018), the Commonsense suite (Bisk et al., 2020; Clark et al., 2019; 2018; Mihaylov et al., 2018; Sakaguchi et al., 2020; Sap et al., 2019; Zellers et al., 2019), MedQA (Jin et al., 2021), CodeFeedback (Zheng et al., 2024) evaluated on HumanEval (Chen et al., 2021), MetaMathQA-40K (Yu et al., 2024) evaluated on GSM8K (Cobbe et al., 2021), and the C4 (Raffel et al., 2020) and PTB (Marcus et al., 1993) language

[1]Source code will be released once the paper is accepted.

Table 4: Comparison of low-rank fine-tuning methods on Microsoft Phi-1.5, Qwen2-1.5B, and LLaMA 3.2-3B models. We compare the accuracy of various layer-wise rank allocation methods. Time denotes the wall-clock time per training epoch.

Model	Method	Param.	MedQA		GSM8K	
			Time	Acc. (%)	Time	Acc. (%)
Microsoft Phi-1.5	Uniform SVD	50%	273 s	31.37	1442 s	24.72
	EigenDamage (Wang et al., 2019)		278 s	32.70	1428 s	24.41
	WeLore (Jaiswal et al., 2025)		253 s	25.24	1170 s	5.61
	Layer-wise Indep. (Guo & Yu, 2025)		281 s	31.29	1450 s	13.80
	MoDeGPT (Lin et al., 2025)		301 s	26.89	1184 s	2.35
	SEA (proposed)		280 s	**33.81**	1447 s	**25.32**
Qwen2-1.5B	Uniform SVD	50%	255 s	30.19	1473 s	17.89
	EigenDamage (Wang et al., 2019)		267 s	30.42	1517 s	14.48
	WeLore (Jaiswal et al., 2025)		291 s	29.72	1226 s	12.28
	Layer-wise Indep. (Guo & Yu, 2025)		271 s	29.87	1533 s	2.43
	MoDeGPT (Lin et al., 2025)		345 s	23.74	1538 s	0.50
	SEA (proposed)		272 s	**31.60**	1540 s	**21.00**

Model	Method	Param.	HumanEval		GSM8K	
			Time	Pass@1	Time	Acc. (%)
LLaMA-3.2-3B	LoRA (Hu et al., 2022)$_{r=8}$	0.38%	3h 15m	25.00	1h 08m	43.37
	AdaLoRA (Zhang et al., 2023)		4h 41m	27.43	1h 20m	43.13
	FlexLoRA (Liu et al., 2026)		3h 46m	32.92	1h 15m	46.55
	IGU-LoRA (Cui et al., 2026)		3h 57m	26.76	1h 15m	35.23
	SEA (proposed)		3h 11m	**35.36**	48m	**48.82**

modeling benchmarks. We use models ranging up to the largest feasible under our GPU resources: OPT-350M (Zhang et al., 2022), Phi-1.5 (Li et al., 2023), Qwen2-1.5B (Yang et al., 2024), LLaMA-3.2-3B (Grattafiori et al., 2024), and LLaMA-2-7B (Touvron et al., 2023). All results are averaged over 3 independent runs. Since the standard deviations were mostly ≤ 0.1, we omit them for brevity.

4.2 Comparative Study

Application to low-rank reparameterization. We evaluate the effectiveness of SEA by applying it to SVD-based low-rank pruning. We vary the total parameter budget to 75%, 50%, and 25% of the original model size and compare the resulting fine-tuning accuracy against low-rank baselines. Table 2 summarizes the fine-tuning results on the GLUE benchmark (OPT-350M) and Commonsense Reasoning tasks (LLaMA-3.2-3B). *Uniform* is a baseline which assigns an identical rank r to all layers, while *Weight-based* determines ranks solely based on the singular energy of the pre-trained weight matrices. For SEA, gradient calibration is performed using the top 10% highest-loss samples from 10% of the total dataset.

Across all GLUE and Commonsense reasoning benchmarks, SEA outperforms both the Uniform and Weight-based settings. The Weight-based approach does not always surpass the Uniform setting, suggesting that pre-trained weight energy alone is insufficient for optimal rank allocation, whereas SEA demonstrates that gradients from downstream data provide crucial guidance for identifying the most informative singular modes. The performance gap widens as the parameter budget becomes more constrained (e.g., at 25%), demonstrating the robustness of our approach.

Application to LoRA. Table 3 applies SEA to LoRA adapters, with its trainable-parameter budget matched to fixed-rank LoRA at $r = 32$ and $r = 8$ (Appendix B). SEA achieves the highest average in all four settings, improving over uniform-rank LoRA by 0.9 and 1.1 points on GLUE and by 2.7 points on Commonsense. It also exceeds full fine-tuning on GLUE at the higher budget and on Commonsense at both budgets.

Comparison to other low-rank fine-tuning methods. Table 4 compares SEA with EigenDamage (Wang et al., 2019), WeLore (Jaiswal et al., 2025), MoDeGPT (Lin et al., 2025), a layer-wise independent rank selection method (Guo & Yu, 2025), AdaLoRA (Zhang et al., 2023), FlexLoRA (Liu et al., 2026), and IGU-LoRA (Cui et al., 2026), all described in Section 2. As proven in Corollary 3, uniform rank is suboptimal for maximizing retained singular energy, and this is consistent with SEA's superiority across all benchmarks under the same parameter budget, while its static allocation keeps training efficiency comparable to uniform-rank settings.

Table 5: LLaMA-2-7B fine-tuning comparison with low-rank adaptation and reparameterization methods, evaluated on C4 (Raffel et al., 2020) and PTB (Marcus et al., 1993) benchmarks.

Method	Param.	C4		PTB	
		Perplexity ↓	Epoch Time	Perplexity ↓	Epoch Time
Full Model	100%	7.34	1379 sec	7.88	637 sec
SVD-LLMv2	50%	23.06	1170 sec	20.71	538 sec
DobiSVD	50%	23.17	1051 sec	18.02	491 sec
SEA (Reparameterization)	50%	**21.82**	1115 sec	**13.98**	508 sec
AdaLoRA	4.5%	7.84	2066 sec	10.28	965 sec
FlexLoRA	4.5%	**7.69**	1555 sec	8.86	702 sec
SEA (LoRA)	4.5%	7.70	1295 sec	**8.60**	600 sec

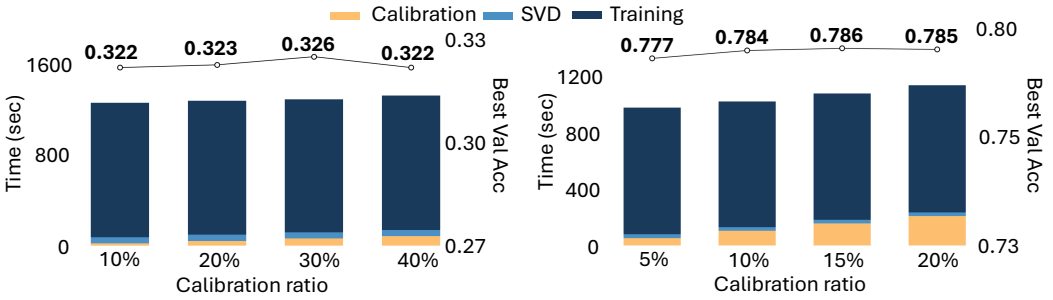

Figure 3: Timing comparisons among calibration, SVD decomposition, and training, with each model compressed to 50% of its parameters. Left: MedQA (Phi-1.5). Right: GLUE (OPT-350M).

Comparison to SOTA adaptive rank methods. Table 5 compares SEA with low-rank reparameterization (SVD-LLMv2, DobiSVD) and adaptation methods (AdaLoRA, FlexLoRA). As a reparameterization method, SEA attains the lowest perplexity on both C4 and PTB, outperforming the next-best method (DobiSVD) by more than perplexity of 4 on PTB, with an epoch time within 7% of DobiSVD. As an adaptation method, SEA (LoRA) matches FlexLoRA on C4 (7.70 vs. 7.69) and achieves the lowest perplexity on PTB (8.60 vs. 8.86), while having the shortest epoch time among the adaptation methods on both datasets.

System efficiency analysis. To evaluate the practicality of SEA, we conduct an ablation study on the required amount of calibration data. Figure 3 presents the results. Accuracy on MedQA and GLUE changes little across calibration ratios, while the one-time SVD accounts for a small fraction of the total time. Another critical observation is that the computational cost increases steadily with the amount of calibration data, without a corresponding gain in accuracy. We find that using $\leq 30\%$ of the training samples with the largest loss values is sufficient to achieve a practical trade-off between accuracy and computational cost.

5 Conclusion

This study addresses the problem of layer-wise rank allocation for pruning modern neural networks, aiming to preserve accuracy while minimizing parameter count. Our importance score exploits the singular energy of gradients to retain the modes most relevant to representing the downstream data. The proposed SEA is a practical rank allocation strategy for fine-tuning that requires neither specialized optimizers nor second-order information, and introduces no additional computational overhead during training. More importantly, SEA is complementary to many existing low-rank pruning and decomposition methods, which typically do not adjust layer-wise rank based on the downstream data distribution.

Limitations and future work – SEA requires a one-time SVD on gradient matrices for calibration, and although performed only once, this cost may become significant for extremely large models. Developing more efficient methods to approximate singular energy, thereby reducing this calibration cost, remains an important direction for future work.

AI USE STATEMENT

We used large language models (LLMs) as writing assistants to polish grammar and phrasing and to check consistency between the text, tables, and appendix. LLMs were not used to generate research ideas, derive the theoretical results, or produce experimental results. All content was verified by the authors, who take full responsibility for it.

REPRODUCIBILITY STATEMENT

The complete proofs of Lemma 1, Theorem 2, Corollary 3, and Proposition 4 are provided in Appendix A. Detailed experimental settings and hyperparameters are provided in Appendix B. Source code will be released once the paper is accepted.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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. 2924–2936, 2019. URL https://aclanthology.org/N19-1300/.

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.

Xuan Cui, Huiyue Li, Run Zeng, Yunfei Zhao, Jinrui Qian, Wei Duan, Bo Liu, and Zhanpeng Zhou. IGU-LoRA: Adaptive rank allocation via integrated gradients and uncertainty-aware scoring. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=MnToYQx9My.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yifan Guo and Alyssa Yu. Integrating independent layer-wise rank selection with low-rank SVD training for model compression: A theory-driven approach. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 5289–5297, 2025. doi: 10.24963/ijcai.2025/589. URL https://www.ijcai.org/proceedings/2025/589.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ajay Kumar Jaiswal, Yifan Wang, Lu Yin, Shiwei Liu, Runjin Chen, Jiawei Zhao, Ananth Grama, Yuandong Tian, and Zhangyang Wang. From low rank gradient subspace stabilization to low-rank weights: Observations, theories, and applications. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26740–26756. PMLR, 2025. URL https://proceedings.mlr.press/v267/jaiswal25a.html.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021. doi: 10.3390/app11146421.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2525–2534. PMLR, 2018. URL https://proceedings.mlr.press/v80/katharopoulos18a.html.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need II: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. MoDeGPT: Modular decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8EfxjTCg2k.

Muqing Liu, Chongjie Si, and Yuheng Jia. FlexLoRA: Entropy-guided flexible low-rank adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tqnkbdYWWm.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

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. URL https://aclanthology.org/D18-1260/.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019. URL https://aclanthology.org/D19-1454/.

Rishi Sonthalia, Michael Murray, and Guido Montúfar. Low rank gradients and where to find them. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=de3kwOXQ9e.

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*, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018. URL https://aclanthology.org/W18-5446/.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. EigenDamage: Structured pruning in the Kronecker-factored eigenbasis. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6566–6575. PMLR, 2019. URL https://proceedings.mlr.press/v97/wang19g.html.

Qinsi Wang, 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*, 2025a. URL https://openreview.net/forum?id=kws76i5XB8.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM V2: Optimizing singular value truncation for large language model compression. 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. 4287–4296, 2025b. URL https://aclanthology.org/2025.naacl-long.217/.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Zhewen Yu and Christos-Savvas Bouganis. SVD-NAS: Coupling low-rank approximation and neural architecture search. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1503–1512, 2023. URL https://openaccess.thecvf.com/content/WACV2023/html/Yu_SVD-NAS_Coupling_Low-Rank_Approximation_and_Neural_Architecture_Search_WACV_2023_paper.html.

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. URL https://aclanthology.org/P19-1472/.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

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.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. OpenCodeInterpreter: Integrating code generation with execution and refinement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12834–12859, 2024. doi: 10.18653/v1/2024.findings-acl.762. URL https://aclanthology.org/2024.findings-acl.762/.

APPENDIX

The appendix is structured as follows:

- Appendix A provides detailed proofs of the proposed lemma, theorem, proposition, and corollary.
- Appendix B summarizes experimental settings corresponding to all the experimental results reported in the main manuscript.
- Appendix C presents additional ablation study and analyses.

A THEORETICAL ANALYSIS

In this appendix section, we present the proofs of the lemmas, theorems, and corollaries. Note that we relax r_l to be continuous and round the solution.

Lemma 1. *For every layer l, the normalized tail energy E_l defined in (2) is convex in r_l. Its piecewise-linear interpolation is convex as well.*

Proof. For any layer $l \in [L]$, the first difference is $E_l(r+1) - E_l(r) = -s_{l,r+1}$. Likewise, $E_l(r) - E_l(r-1) = -s_{l,r}$. Then, for $1 \le r \le d_l - 1$, the second difference is $(E_l(r+1) - E_l(r)) - (E_l(r) - E_l(r-1)) = s_{l,r} - s_{l,r+1}$. Since the normalized energy is ordered in non-increasing order, the second difference $s_{l,r} - s_{l,r+1}$ is always non-negative. Therefore, $E_l(r)$ is convex in r. In addition, the piecewise-linear interpolation of E_l has slope $-s_{l,r+1}$ on the segment $(r, r+1)$, so the second difference computed above is precisely the increment of this slope. Since that increment is non-negative, the slopes are non-decreasing and the interpolation is convex. □

Theorem 2. *The global optimal rank allocation $\mathbf{r}^*$ across L layers that minimizes the normalized tail energy $\sum_l E_l(r_l)$ under a fixed parameter budget $\sum_l c_l r_l = P_{budget}$ satisfies:*

$$\exists\, \lambda \ge 0 \quad \text{s.t.} \quad \frac{s_{l,\, r_l^*+1}}{c_l} \le \lambda \le \frac{s_{l,\, r_l^*}}{c_l} \quad \text{for all } l. \tag{11}$$

Here, r_l is relaxed to be continuous. At a layer whose optimal rank is not integral, the two bounds in (3) coincide.

Proof. By applying a Lagrangian multiplier λ to the budget constraint, we have

$$\mathcal{L}(\mathbf{r}, \lambda) \;=\; \sum_l E_l(r_l) \;+\; \lambda \left(\sum_l c_l r_l - P_{budget} \right).$$

By Lemma 1, E_l is convex, and the constraint is affine. The problem is therefore convex, and a stationary point of $\mathcal{L}$ is a global optimum.

Taking the derivative with respect to r_l requires care, because E_l is piecewise linear and is not differentiable at integral ranks. As shown in the proof of Lemma 1, the slope of E_l on the segment $(r-1, r)$ equals $-s_{l,r}$. The one-sided derivatives of $\mathcal{L}$ at r_l are therefore

$$\frac{\partial^- \mathcal{L}}{\partial r_l} = -\, s_{l,r_l} + \lambda c_l, \qquad \frac{\partial^+ \mathcal{L}}{\partial r_l} = -\, s_{l,r_l+1} + \lambda c_l.$$

At the optimum, decreasing r_l must not decrease $\mathcal{L}$, and neither must increasing it, which gives

$$\frac{\partial^- \mathcal{L}}{\partial r_l} \le 0 \qquad \text{and} \qquad \frac{\partial^+ \mathcal{L}}{\partial r_l} \ge 0.$$

Substituting the one-sided derivatives and rearranging the terms, we obtain $s_{l,r_l^*+1} \le \lambda c_l \le s_{l,r_l^*}$, and thus:

$$\frac{s_{l,r_l^*+1}}{c_l} \le \lambda \le \frac{s_{l,r_l^*}}{c_l}. \tag{12}$$

Table 6: Hyper-parameter settings for experiments shown in Tables 2 and 3. Top: common fine-tuning hyperparameters. Bottom: LoRA configurations used in Table 3; fixed-rank LoRA baselines use ranks 32 and 8 with LoRA scaling factors $\alpha = 64$ and 16, and SEA uses model-specific trainable-parameter budgets matched to the trainable parameter scale of each fixed-rank setting.

Model	Dataset	Method	Batch	Accum.	LR	Epochs	Seq. Len.
OPT-350M	GLUE	MNLI	64	-	1e-4	4	128
		QQP			1e-5		
		QNLI			1e-5		
		SST-2			1e-5		
		CoLA			1e-5		
		STS-B			1e-5		
		MRPC			1e-5		
		RTE			1e-5		
LLaMA-3.2-3B	Commonsense	BoolQ	4	16	2e-4	3	512
		PiQA			2e-4		
		HellaS.			2e-4		
		Wino.			2e-4		
		SIQA			2e-4		
		ARC-e.			2e-4		
		ARC-c.			2e-4		
		OBQA			2e-4		

Model	Setting	LoRA Rank	LoRA α	SEA Param. Budget
OPT-350M	High budget	32	64	3.66%
	Low budget	8	16	0.94%
LLaMA-3.2-3B	High budget	32	64	1.51%
	Low budget	8	16	0.38%

Whenever r_l^* is not integral the two one-sided derivatives coincide, and the condition reduces to the usual stationarity $\partial\mathcal{L}/\partial r_l = 0$, which pins λ to the single value $s_{l,\lceil r_l^* \rceil}/c_l$. At a layer with $r_l^* = 0$ only the right derivative applies, and at a layer with $r_l^* = d_l$ only the left one. The multiplier λ is shared across layers because it is attached to the single budget constraint, which yields (3). □

Corollary 3. *Let $r^{uni} = \lfloor P_{budget} / \sum_l c_l \rfloor$ denote the uniform rank. Then, $\sum_l E_l(r^{uni}) \geq \sum_l E_l(r_l^*)$, and the inequality is strict unless (3) holds at r^{uni}, that is, unless some single λ satisfies $s_{l,r^{uni}+1}/c_l \leq \lambda \leq s_{l,r^{uni}}/c_l$ in every layer.*

Proof. The uniform allocation satisfies $\sum_l c_l r^{uni} \leq P_{budget}$ and is therefore feasible, which gives the inequality since $\mathbf{r}^*$ is a minimizer. If no λ satisfies (3) at r^{uni}, then r^{uni} violates a necessary condition for optimality and the inequality is strict. □

Proposition 4. *Let x^{opt} be an optimal solution of* (5) *and let $S(x) := \sum_{l,i} s_{l,i} x_{l,i}$. Assume that the densities $s_{l,i}/c_l$ are distinct and let (l_b, i_b) denote the discarded mode of the largest density. Then*

$$S(x^{\text{opt}}) - S(x^*) \leq \frac{\Delta}{c_{l_b}} s_{l_b,i_b} < s_{l_b,i_b}. \tag{13}$$

Proof. Consider the continuous relaxation of (5) obtained by replacing $x_{l,i} \in \{0,1\}$ with $x_{l,i} \in [0,1]$, and let S_{LP} be its optimal value. The relaxed problem is a fractional knapsack problem, and its optimum is attained by selecting the modes in decreasing order of the density $s_{l,i}/c_l$ until the budget is exhausted, where only the last selected mode may be fractional. By construction, x^* selects exactly the modes whose density exceeds λ^*, and (l_b, i_b) is the next mode in this order. The relaxed optimum therefore agrees with x^* on all other modes and spends the residual budget Δ on mode (l_b, i_b), which gives

$$S_{\text{LP}} = S(x^*) + \frac{\Delta}{c_{l_b}} s_{l_b,i_b}.$$

Table 7: Hyper-parameter settings for experiments shown in Table 4.

Model	Dataset	Method	Epochs	Batch	Accum.	LR	Seq. Len.
Phi-1.5	MedQA	Uniform	5	4	16	1e-4	512
		EigenDamage				1e-4	
		WeLore				1e-4	
		MoDeGPT				1e-4	
		Layer-wise Independent				2e-4	
		SEA				1e-4	
	MetaMathQA	Uniform	3			1e-4	
		EigenDamage				1e-4	
		WeLore				2e-4	
		MoDeGPT				2e-4	
		Layer-wise Independent				2e-4	
		SEA				1e-4	
Qwen2-1.5B	MedQA	Uniform	5	4	16	1e-4	512
		EigenDamage				1e-4	
		WeLore				1e-4	
		MoDeGPT				1e-4	
		Layer-wise Independent				2e-4	
		SEA				1e-4	
	MetaMathQA	Uniform	3			1e-4	
		EigenDamage				1e-4	
		WeLore				2e-4	
		MoDeGPT				2e-4	
		Layer-wise Independent				2e-4	
		SEA				1e-4	
LLaMa-3.2-3B	CodeFeedBack	LoRA	1	4	8	5e-5	512
		AdaLoRA				5e-4	
		FlexLoRA				2e-4	
		IGU-LoRA				1e-3	
		SEA				5e-4	
	MetaMathQA	LoRA				5e-4	
		AdaLoRA				5e-3	
		FlexLoRA				1e-3	
		IGU-LoRA				1e-2	
		SEA				1e-3	

Since the feasible set of (5) is contained in that of its relaxation, $S(x^{\text{opt}}) \leq S_{\text{LP}}$. It remains to show $\Delta < c_{l_b}$. If $\Delta \geq c_{l_b}$, then adding mode (l_b, i_b) to x^* would still respect the budget, so any threshold slightly below the density of (l_b, i_b) would be admissible, contradicting the minimality of λ^*. □

B Detailed Experimental Settings

In this appendix section, we summarize the experimental settings corresponding to all the empirical results we reported in the main manuscript, including dataset splits, implementation details, hyperparameter configurations, and algorithm-specific settings. The random seed was not fixed across runs, in order to better reflect the variance observed under realistic training conditions. We did not set the seed explicitly; each run used the default, non-deterministic initial seed of PyTorch.

B.1 Implementation details

We conduct experiments reported in Tables 2, 3, 4 and 5 on servers equipped with NVIDIA RTX 4090 GPUs. Our implementation is based on PyTorch 2.10.0 (built with CUDA 12.8) with NVIDIA driver 550.144.03 (CUDA 12.4). Tables 6, 7, 8, and 9 summarize the fine-tuning hyperparameters, and rank or parameter-budget configurations used in these experiments. Unless otherwise specified, we use AdamW as the optimizer with momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$ and weight

Table 8: Algorithm-specific hyperparameters for the low-rank reparameterization and LoRA-based fine-tuning experiments shown in Table 4.

Model	Dataset	Method	Algorithm-specific hyperparameters
Phi-1.5	MedQA	Uniform SVD	rank = 565
		EigenDamage	threshold = 0.7
		WeLore	$k = 0.231$
		MoDeGPT	ϕ_{avg} = 0.12, $\lambda = 1.0$, $\epsilon = 0.1$, calib num = 64
		Layer-wise Indep.	ϵ = 2.07e-3
		SEA	ratio = 0.36
Phi-1.5	MetaMathQA	Uniform SVD	rank = 565
		EigenDamage	threshold = 0.65
		WeLore	$k = 0.231$
		MoDeGPT	ϕ_{avg}= 0.12, $\lambda = 1.0$, $\epsilon = 0.1$, calib num = 64
		Layer-wise Indep.	ϵ = 9.38e-4
		SEA	ratio = 0.36
Qwen2-1.5B	MedQA	Uniform SVD	rank = 480
		EigenDamage	threshold = 0.7
		WeLore	$k = 0.328$
		MoDeGPT	ϕ_{avg} = 0.5, $\lambda = 1.0$, $\epsilon = 0.1$, calib num = 64
		Layer-wise Indep.	ϵ = 1.03e-3
		SEA	ratio = 0.35
Qwen2-1.5B	MetaMathQA	Uniform SVD	rank = 480
		EigenDamage	threshold = 0.57
		WeLore	$k = 0.328$
		MoDeGPT	ϕ_{avg} = 0.5, $\lambda = 1.0$, $\epsilon = 0.1$, calib num = 64
		Layer-wise Indep.	ϵ = 8.5e-4
		SEA	ratio = 0.35
LLaMA-3.2-3B	CodeFeedback	LoRA	–
		AdaLoRA	init $r = 16$, target $r = 8$, Tinit = 400, TFinal = 200, step = 10, $\beta = (0.85, 0.85)$
		FlexLoRA	init $r = 8$, $b = 4$, step = 20
		IGU-LoRA	init $r = 16$, target $r = 8$, Tinit = 300, TFinal = 200, step = 20, $\beta = (0.85, 0.85)$
		SEA	ratio = 0.003
LLaMA-3.2-3B	MetaMathQA	LoRA	–
		AdaLoRA	init $r = 16$, target $r = 8$, Tinit = 400, TFinal = 200, step = 10, $\beta = (0.85, 0.85)$
		FlexLoRA	init $r = 8$, $b = 2$, step = 20
		IGU-LoRA	init $r = 16$, target $r = 8$, Tinit = 300, TFinal = 200, step = 20, $\beta = (0.85, 0.85)$
		SEA	ratio = 0.003

decay 1×10^{-2}. We use a linear learning rate scheduler with warmup ratio 3×10^{-2} and set the maximum gradient norm to 1.0. For each model, dataset, and method setting, we tune the learning rate over a predefined candidate set and use the value that achieves the best validation performance. The selected learning rates and other dataset-specific hyperparameters, including batch size, gradient accumulation steps, number of epochs, and maximum sequence length, are reported in the corresponding hyperparameter tables. For the SVD-based low-rank reparameterization experiments in Table 2, we evaluate uniform rank allocation under three parameter-retention ratios: 75%, 50%, and 25%. In this setting, a single shared rank is assigned to the target weight matrices of each model so that the compressed model satisfies the corresponding parameter-retention budget. For OPT-350M, we use uniform ranks of 495, 310, and 123 for the 75%, 50%, and 25% retention settings, respectively. For LLaMA 3.2-3B, the corresponding uniform ranks are 1426, 800, and 269. For `SEA` in the low-rank reparameterization setting, the rank allocation is determined by the `SEA` score under the same target parameter-retention budget. The `SEA` ratio controls the fraction of retained low-rank parameters assigned by our allocation rule, and it is selected to satisfy the corresponding parameter

budget for each model and retention setting. Thus, all compared allocation schemes use the same parameter-retention ratio, while differing only in how ranks are distributed across layers.

For the LoRA fine-tuning experiments, SEA uses model-specific trainable-parameter budgets matched to fixed-rank LoRA with ranks 32 and 8, as listed in the bottom panel of Table 6. For a fair comparison, methods compared within the same model, dataset, and budget setting share the optimizer, scheduler, warmup configuration, batch configuration, number of epochs, and evaluation protocol. We fine-tune on the official training splits and report results on the validation sets for GLUE and the Commonsense datasets (BoolQ, PiQA, HellaSwag, WinoGrande, SIQA, ARC-easy, ARC-challenge, OpenBookQA), and on the dev set of the official 4-option USMLE split for MedQA (10,178 train / 1,272 dev / 1,273 test). The GSM8K and HumanEval results in Table 4 are obtained on their test sets after fine-tuning on MetaMathQA-40K and CodeFeedback, respectively. MedQA is scored without generation by selecting the answer option with the lowest mean token-level negative log-likelihood, and GSM8K by greedy decoding of up to 256 new tokens and comparing the final numeric answer with the reference.

B.2 Hyperparameter settings for reparameterization method comparison

Tables 7 and 8 summarize the hyperparameter settings used for the reparameterization method comparison experiments reported in Table 4. Table 7 reports the common fine-tuning hyperparameters, while Table 8 lists the algorithm-specific hyperparameters for each reparameterization or adaptation method. These experiments compare both fine-tuning accuracy and epoch time across different layer-wise rank allocation methods.

For Microsoft Phi-1.5 and Qwen2-1.5B, we evaluate SVD-based low-rank reparameterization methods under a 50% parameter-retention budget. The compared methods include Uniform SVD, EigenDamage, WeLore, the Layer-wise Independent rank selection method, MoDeGPT, and SEA.

The selected learning rates, batch sizes, gradient accumulation steps, numbers of epochs, and maximum sequence lengths are reported in Table 7. Method-specific hyperparameters, such as the uniform rank, EigenDamage threshold, WeLore's normalized singular-value threshold k, MoDeGPT calibration settings, and the Layer-wise Independent loss-error tolerance ϵ, are reported separately for each method. The epoch time reported in Table 4 denotes the elapsed real-world training time required to complete one fine-tuning epoch under the corresponding method and dataset.

For WeLore, we adopt only its rank-allocation rule, which discards the singular directions whose min-max normalized singular value falls below a global threshold k and factorizes a matrix only if more than half of its directions are discarded. Since WeLore specifies its budget as a rank reduction rather than a parameter count, we search k so that the compressed model retains 50% of the parameters (Table 8). As for the other methods, the selected matrices are factorized by truncated SVD and all parameters are fine-tuned, rather than training only the low-rank factors as in the original WeLore.

B.3 Hyperparameter settings for LoRA experiments

LoRA experiments presented in Table 4 are configured according to Tables 7 and 8. We used AdamW optimizer with momentum 0.9 and Cosine learning rate scheduler with weight decay 1e-2, beta values (0.9, 0.999). Through grid search performed at least twice per setting, we tuned learning rate among 5e-3, 1e-3, 1e-4, 2e-4, 5e-5, 1e-5. We used LoRA rank 8 and LoRA scaling factor α = 16 for LLaMA 3.2-3B, we evaluated LoRA-based fine-tuning methods under a 0.38% trainable parameter budget, including fixed-rank LoRA, AdaLoRA, FlexLoRA, IGU-LoRA, and SEA. Algorithm specific hyperparameters (e.g., AdaLoRA rank schedule, FlexLoRA block setting, and SEA ratio) are reported separately in Table 8.

B.4 Hyperparameter settings for the LLaMA-2-7B comparison

Table 9 summarizes the hyperparameter settings used for the experiments reported in Table 5, which compares SEA with low-rank adaptation methods (AdaLoRA, FlexLoRA) and low-rank reparameterization methods (SVD-LLMv2, DobiSVD) on LLaMA-2-7B. For the reparameterization setting, all methods are evaluated under a 50% parameter-retention budget, and the layer-wise rank allocation of SEA is determined by the SEA score under the same retention budget. For the adaptation setting, all methods are evaluated under a 4.5% trainable-parameter budget. Models are fine-tuned

Table 9: Hyper-parameter settings for the LLaMA-2-7B experiments shown in Table 5. Top: common fine-tuning hyperparameters. Bottom: algorithm-specific hyperparameters.

Model	Dataset	Method	Epochs	Batch	Accum.	LR	Seq. Len.
LLaMA-2-7B	C4	AdaLoRA	3	1	8	1e-3	512
		FlexLoRA				1e-3	
		SEA (LoRA)				1e-4	
		SVD-LLMv2				5e-5	
		DobiSVD				5e-5	
		SEA (Reparameterization)				5e-5	
	PTB	AdaLoRA	3			1e-3	
		FlexLoRA				1e-3	
		SEA (LoRA)				2e-4	
		SVD-LLMv2				5e-5	
		DobiSVD				5e-5	
		SEA (Reparameterization)				5e-5	

Setting	Method	Algorithm-specific hyperparameters
LoRA (≈4.5%)	AdaLoRA	init $r = 32$, target $r = 16$, Tinit = 400, TFinal = 200, step = 10, $\beta = (0.85, 0.85)$
	FlexLoRA	init $r = 16$, $b = 2$, step = 10
	SEA (LoRA)	ratio = 0.045
Reparameterization(≈50%)	SVD-LLMv2	ratio = 0.50, whitening nsamples = 256, direct-U-update = False
	DobiSVD	ratio = 0.50, $\beta = 10.0$, γ-lr = 5.0, γ-steps = 1200, $\lambda_{\text{compress}} = 200$, $\lambda_{\text{value}} = 1.0$, IPCA samples = 128
	SEA (Reparameterization)	ratio = 0.50

and evaluated on the C4 and PTB language modeling benchmarks, and we report validation perplexity together with the epoch time, which denotes the elapsed real-world training time required to complete one fine-tuning epoch under the corresponding method and dataset. For PTB, we use the standard Mikolov-preprocessed corpus (`ptb.train.txt` and `ptb.valid.txt`) in its entirety without subsampling; the training split contains 887,521 words in 42,068 sentences. For C4, we stream the official English corpus (`allenai/c4`, `en`) and use the first 4,800 training documents and 512 validation documents, which is a small subset of the full corpus. For both datasets, the tokenized text is concatenated and segmented into non-overlapping blocks of 512 tokens, and each block is treated as a single training sample (batch size 1 with gradient accumulation over 8 steps). This yields 2,385 training and 188 validation blocks (≈1.22M and ≈0.10M tokens) for PTB, and 5,208 training and 549 validation blocks (≈2.67M and ≈0.28M tokens) for C4. The common fine-tuning configuration is reported in the top panel of Table 9. Algorithm-specific hyperparameters, such as the AdaLoRA rank schedule, the FlexLoRA block setting, and the `SEA` ratio, are reported in the bottom panel of Table 9.

C Additional Ablation Study

In this appendix section, we provide additional ablation study results and analyses.

Gradient calibration metric – Table 10 presents the GLUE fine-tuning results with OPT-350M using different gradient calibration strategies. The model is pruned using SVD to retain 25% of the parameters. Across all sub-tasks, the best accuracy is achieved when calibrating gradients using large-loss samples. That is, large-loss samples more clearly reveal which layers are critical to accuracy, and selectively calibrating gradients on these samples enables more effective utilization of the parameter budget.

Layer-wise ranks in weights and gradients – In addition to the heatmap we presented in the main manuscript, we also show the same layer-wise ranks of OPT-350M during fine-tuning on GLUE benchmark datasets. Figure 4 shows the results. First, the ranks of both weights and gradients do not dramatically change throughout the fine-tuning. Furthermore, the ranks of gradients lie within

Table 10: GLUE (OPT-350M) performance comparison across different gradient calibration settings (25% parameters, learning rate 1e-4, batch size 64). "Small Loss" and "Large Loss" denote calibration using the lowest-loss and highest-loss samples, respectively.

Method	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
Random	73.89	86.99	82.43	86.96	55.48	49.98	71.32	53.67	70.09
Small Loss	73.89	87.30	81.74	87.12	54.88	41.47	71.24	**54.75**	69.05
Large Loss	**75.02**	**87.62**	**82.76**	**87.65**	**69.32**	**67.42**	**72.39**	54.39	**74.57**

smaller values than those of weights. These observations are well aligned with those discussed in Motivational empirical study section.

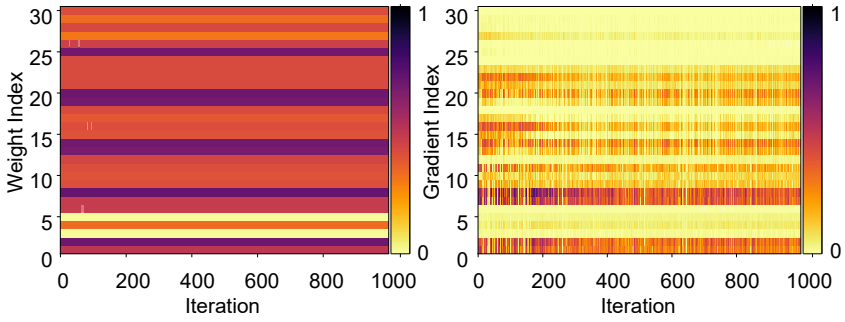

Figure 4: Normalized layer-wise ranks capturing 90% of the singular value energy. The left heatmap shows its evolution across 30 weight tensors of the OPT-350M during fine-tuning on the GLUE benchmark, while the right one shows the corresponding trends for the gradients. The y-axis represents individual weight tensor indices, and the x-axis represents fine-tuning iterations. The energy distribution remains stable throughout fine-tuning in both weights and gradients.

Figure 5 shows the layer-wise singular values after one epoch of fine-tuning. We use OPT-350M trained on the GLUE dataset. A key observation is that layers exhibit noticeably different singular value distributions for weights and gradients. For example, in layer 28, the principal singular values of the gradients are much larger than those of the weights, whereas the weights exhibit larger singular values in other parts of the spectrum. These results provide strong support for our proposed gradient-based, layer-wise singular energy-aware rank allocation strategy.

Singular-mode selection order – SEA allocates rank by greedily retaining the singular modes with the *highest* efficiency ratio $s_{l,i}/c_l$ (descending order) until the parameter budget is exhausted. To isolate how much of SEA's improvement is attributable to this ordering choice, as opposed to simply having a layer-wise (rather than uniform) rank budget, we run an inverted ablation that instead retains the *lowest*-ranked modes first (ascending order), under the identical parameter budget, model, and fine-tuning protocol as SEA at ≈50% retention (LLaMA-2-7B). Figure 6 reports the resulting perplexity on C4 and PTB. Reversing the selection order is catastrophic: perplexity increases by 7.7× on C4 (21.82 → 168.69) and 3.6× on PTB (13.98 → 50.01). These results confirm that SEA's gains come specifically from correctly identifying *which* modes to keep via gradient singular energy, not merely from having a non-uniform, layer-wise rank budget.

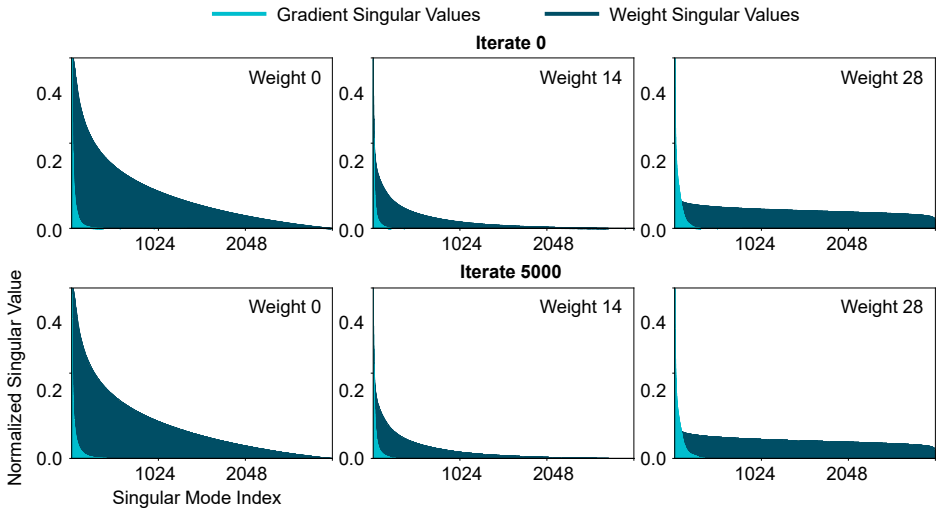

Figure 5: Comparison of singular value distributions in OPT-350M during fine-tuning. A key observation is that layers exhibit markedly different singular value distributions for weights and gradients.

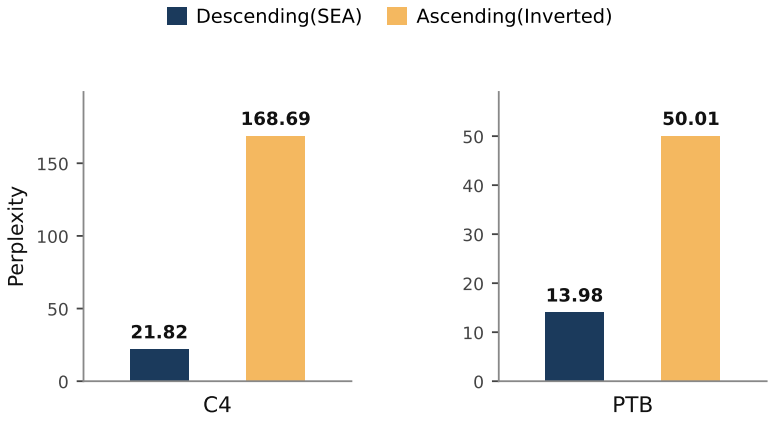

Figure 6: Effect of singular-mode selection order on SEA (Reparameterization) at ≈50% parameter retention (LLaMA-2-7B). *Descending* retains the highest-efficiency-ratio modes first (the proposed SEA rule); *Ascending* inverts this and retains the lowest-efficiency-ratio modes first. Lower perplexity is better.