Low-Rank Compression of Language Models via Differentiable Rank Selection

Sidhant Sundrani Francesco Tudisco Pasquale Minervini
University of Edinburgh, Edinburgh, UK
sidhantls@outlook.com {f.tudisco, p.minervini}@ed.ac.uk

Abstract

Approaches for compressing large-language models using low-rank decomposition have made strides, particularly with the introduction of activation and loss-aware SVD, which improves the trade-off between decomposition rank and downstream task performance. Despite these advancements, a persistent challenge remains–selecting the optimal ranks for each layer to jointly optimise compression rate and downstream task accuracy. Current methods either rely on heuristics that can yield sub-optimal results due to their limited discrete search space or are gradient-based but are not as performant as heuristic approaches without post-compression fine-tuning. To address these issues, we propose Learning to Low-Rank Compress (LLRC), a gradient-based approach which directly learns the weights of masks that select singular values in a fine-tuning-free setting. Using a calibration dataset, we train only the mask weights to select fewer and fewer singular values while minimising the divergence of intermediate activations from the original model. Our approach outperforms competing ranking selection methods that similarly require no post-compression fine-tuning across various compression rates on common-sense reasoning and open-domain question-answering tasks. For instance, with a compression rate of 20% on Llama-2-13B, LLRC outperforms the competitive Sensitivity-based Truncation Rank Searching (STRS) on MMLU, BoolQ, and OpenbookQA by 12%, 3.5%, and 4.4%, respectively. Compared to other compression techniques, our approach consistently outperforms fine-tuning-free variants of SVD-LLM and LLM-Pruner across datasets and compression rates. Our fine-tuning-free approach also performs competitively with the fine-tuning variant of LLM-Pruner.

Keywords: LLM Compression, Low-Rank Decomposition, Rank Selection

1. Introduction

Large language models (LLMs) such as GPT-3 (Brown et al., 2020) and LLaMA (Touvron et al., 2023) show remarkable results in natural language understanding and generation tasks. These models are not just pivotal in zero-shot language modelling but also extend their utility to applications such as code generation (Chen et al., 2021), conversational agents (Kumar et al., 2023), and personalised education (Kasneci et al., 2023). Despite the success of these models in solving a wide range of tasks, their use is limited by high computing and memory requirements. For example, LLaMA-2 comes in 7 billion and 40 billion parameter variants (Touvron et al., 2023), requiring 25.79 GB and 153.87 GB of memory, respectively. As these models grow, various compression techniques have been developed to reduce their size.

Quantisation, which reduces the number of bits required to represent each parameter, is widely used to compress language models (Dettmers et al., 2022; Frantar et al., 2023; Lin et al., 2024). As an alternative to quantisation, other works explored the structural pruning of LLMs (Ma et al., 2023; Xia et al., 2024). These works follow two stages for compression: first, pruning for model compression and then a training stage to recover performance (Ma et al., 2023; Xia et al., 2024).

In addition to quantisation and pruning, recent works show that applying low-rank matrix decomposition techniques can also compress and reduce the memory requirements of language models (Hsu et al., 2022a; Li et al., 2023; Yuan et al., 2024; Wang et al., 2025; Wong et al., 2025).

Recent methods have further refined low-rank compression through weighted or activation-aware singular value decomposition, a variant that makes the decomposition loss-aware or activation-aware (Hsu et al., 2022a; Yuan et al., 2024; Gao et al., 2024; Wong et al., 2025). Nevertheless, a persistent challenge across these low-rank approaches is the selection of optimal layer-wise ranks, which ultimately governs the achievable balance between parameter reduction and model fidelity.

Given that research has shown that different layers of a language model may have different optimal compression rates (Yuan et al., 2024; Nawrot et al., 2024), using a constant rank across layers may not be an effective solution. To address this problem, Yuan et al. (2024) proposes a heuristic named Sensitivity-based Truncation Rank Searching (STRS), which iteratively searches for optimal ranks per layer by evaluating model perplexity on a small calibration set. Despite proving better than a naive selection of constant compression ratios across layers, this approach has two core problems that can lead to suboptimal solutions. First, the search space of ranks is a discrete set of only 10 elements, significantly restricting the number of options available for the ranks. Second, the

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 10031–10045
11–16 May 2026.

optimal decomposition of each layer is identified independently, without taking into account the decomposition of the other layers.

On the other hand, (Gao et al., 2024) proposed a rank selection approach to learn the optimal ranks through gradient descent. In Adaptive Rank Selection (ARS), a binary masking mechanism is used for optimising the number of ranks through training (Gao et al., 2024). Using GRUs (Dey and Salem, 2017) and linear projections, ARS introduced a learnable singular value masking layer into the SVD reconstruction from which the rank was extracted. However, a key shortcoming is that rank selection using ARS leads to heavy performance degradation and requires an expensive post-compression training stage. Moreover, the work does not explore optimal methods to convert the learned mask into a final low-rank model. Similar to this approach, pruning techniques as (Wang et al., 2020b) have also explored learning singular value masking for compression; however, unlike ARS, it performs training of the entire model and focuses on smaller models like BERT. To address these problems in rank selection in low-rank decomposed models, we propose a method called *Learning to Low-Rank Compress* (LLRC).

LLRC can learn the optimal per-layer factorisation ranks by introducing a singular value selection mask Σ_m into the matrix reconstruction, which is optimised via gradient-based optimisation; Fig. 1 provides a high-level outline of the method. The mask Σ_m is trained using a multi-objective loss function that enables the balancing of compression costs and downstream task performance. This training approach is lightweight, as it only requires gradient computation for the linear singular value masking layers rather than for the entire model. Following training, after the overall desired compression rate is achieved, the masks are directly utilised to select the optimal number of singular values. We also define an optimal approach to handle effectively uncompressed layers after training to improve performance. To summarise, our key contributions are the following:

- A fine-tuning free technique for LLMs called *Learning to Low-Rank Compress (LLRC)* to learn optimal singular values for each layer through training on a small calibration dataset.
- A learnable singular value masking linear parameter that learns, in a fine-tuning-free setting, to select the most optimal *any-k* singular values for compression of LLMs.

2. Related Work

Model compression is a crucial field in deep learning that focuses on reducing the computational costs associated with deploying models while maintaining their performance. There are several approaches to neural network compression, including pruning (Han et al., 2015; Zhu and Gupta, 2017; Ma et al., 2023), quantisation (Bai et al., 2021; Xiao et al., 2024), and low-rank factorisation, the focus of this work. Specifically in natural language processing (NLP), there have been various efforts along these lines. Early work aimed to reduce the number of parameters in LSTMs. For instance, (Winata et al., 2019) applied low-rank decomposition techniques such as Semi-NMF and SVD for LSTM compression and compared the results to pruning. In contrast to performing low-rank factorisation on a trained model, other works applied tensor decomposition to re-parameterise the model architecture for training (Yang et al., 2017; Pan et al., 2019; Zangrando et al., 2023).

More recent works focused on applying low-rank factorisation to compress language models without full re-training (Yuan et al., 2024; Hsu et al., 2022a; Wang et al., 2025; Wong et al., 2025). Rather than minimising the reconstruction error of the weight matrix directly, *weighted* SVD methods incorporate additional information—such as gradient statistics or input activations—to re-weight the decomposition so that components more influential to downstream behaviour are preserved. For instance, (Hsu et al., 2022a) develops a weighted singular value decomposition approach called Fisher-Weighted SVD (FWSVD) that uses Fisher information to preserve model performance on a given downstream task, outperforming naive SVD for compression (Hsu et al., 2022a). Similarly, Activation-Aware SVD (ASVD) incorporates intermediate activations into the decomposition, optimising for output reconstruction rather than weight approximation (Yuan et al., 2024). Instead of minimising the output loss of a single transformation, A3 minimises the output loss of a group of transformations, formulating low-rank decompositions across groups of weights (Wong et al., 2025).

In such approaches, the singular value rank for each layer has to be determined. A relevant approach is Sensitivity-based Truncation Rank Searching (STRS; Yuan et al., 2024), which iteratively searches for optimal ranks per layer by evaluating model perplexity on a small calibration set. STRS performs a binary search over a discrete set of 10 pre-defined compression rates, applying low-rank decomposition to one layer at a time while leaving the rest of the network unchanged (Yuan et al., 2024). ARS (Gao et al., 2024) instead tackles rank selection via gradient-based optimisation, employing a recurrent neural network and linear layers to predict binary masks over singular values, but requires an expensive post-compression fine-tuning stage to recover performance. SVD-LLM takes a complementary direction, focusing on

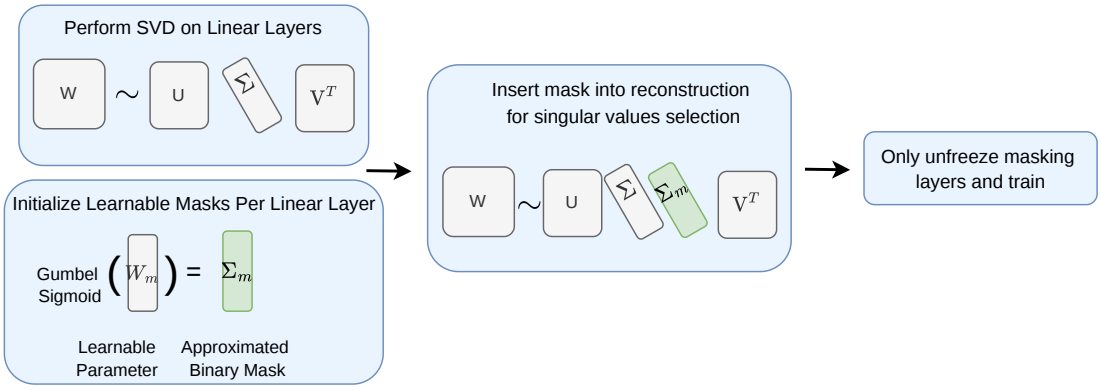

Figure 1: Outline of the training process of our proposed method to learn SVD ranks for low-rank compression

making the decomposition truncation-aware rather than learning per-layer ranks (Wang et al., 2025): it introduces truncation-aware data whitening to establish a direct correlation between singular value magnitude and layer output error, enabling efficient compression without a per-layer rank search.

3. Background

We now describe how Singular Value Decomposition (SVD) can be used to compress linear layers. Moreover, we also briefly cover details of ASVD (Yuan et al., 2024), a weighted SVD technique which yields higher model compression rates at lower performance loss.

3.1. Compressing Transformers with SVD

SVD is a fundamental matrix factorisation technique used in various fields, including image processing and machine learning (Chen, 2018; Yang, 2021). The SVD decomposes a matrix W into the product of three other matrices: U_r, Σ_r, and V_r^T as shown in Eq. (1):

$$W = U_r \Sigma_r V_r^T \tag{1}$$

Here, $W \in \mathbb{R}^{m \times n}$ is a rank-r matrix, $U_r \in \mathbb{R}^{m \times r}$ and $V_r \in \mathbb{R}^{n \times r}$ are orthogonal, $\Sigma_r \in \mathbb{R}^{r \times r}$ is the diagonal matrix of the singular values, and r is the rank of the decomposition.

Decreasing r reduces the dimension of U_r and V_r but worsens the approximation of W. In particular, for any $k \leq r$, it holds:

$$U_k \Sigma_k V_k^T = \arg \min_{\text{rank}(W_k)=k} \|W_k - W\|_2 \,.$$

This would result in the storage of three smaller matrices, U, Σ, and V, in place of a larger matrix W. Therefore, if W is of shape (m, n), the compression can be quantified by the parameter ratio in Eq. (2), where r denotes the rank of decomposition:

$$\text{Param Ratio} = \frac{r(m+n)}{mn}. \tag{2}$$

Such an approximation can be applied to linear layers in a transformer, such as the query projections and linear projections in the feed-forward network.

3.2. Activation-Aware SVD

Activation-Aware SVD is a form of weighted SVD that aims to minimise the reconstruction error of the output of a linear transformation rather than minimising the error of the reconstructed weight matrix (Yuan et al., 2024). This approach can be formulated as optimising the following quantity:

$$\arg \min_{\text{rank}(W_k)=k} \|W_k X - WX\|_2 \tag{3}$$

where W_k is the reconstructed weight using k singular values and X is an input into the linear transformation. This is achieved by performing SVD on WS rather than W and then scaling the reconstructed weight by S^{-1}, where S is a matrix calculated from a set of inputs X designed to capture the influence of the input channels on the weights (Yuan et al., 2024).

4. Learning to Low-Rank Compress

Learning to Low-Rank Compress (LLRC) centres on applying SVD to each weight matrix targeted for compression, followed by a learnable masking layer that selects singular values, as shown in Fig. 1.

During training, these masking layers can adaptively learn highly different compression rates for each layer. Training jointly optimises compression and model performance using a multi-task training objective The only learnable parameters are the weights that control the masking, while the rest of the model is frozen. The following sub-sections further provide details of the training procedure.

4.1. Applying SVD

We perform SVD on all linear projection layers in the model except for the logits layer. Given the ability of

weighted SVD approaches to retain performance at higher compression rates (Hsu et al., 2022b; Yuan et al., 2024), we utilise ASVD as a drop-in replacement for SVD in most of our experiments. For consistency with (Yuan et al., 2024), we use $\alpha = 0.5$.

4.2. Trainable Singular Value Selection

Following decomposing a linear layer into its factors using SVD as in Eq. (1), we introduce a learnable mask inserted in its reconstruction to select singular values, outlined in Fig. 1 and is defined as follows:

$$W = U\Sigma(\Sigma_{\text{mask}})V^T, \Sigma_{\text{mask}} = g(W_{\text{learnable}}). \quad (4)$$

The matrix $W_{\text{learnable}} \in \mathbb{R}^{1\times\text{rank}}$ is a learnable parameter, used to generate Σ_{mask}. The learnt mask is represented by $\Sigma_{\text{mask}} \in \{0,1\}^{1\times\text{rank}}$ and the reconstructed weight is $W \in \mathbb{R}^{m\times n}$. The function g in Eq. (4) represents Gumbel-Sigmoid (Jang et al., 2017). Gumbel-Sigmoid re-parameterises the Bernoulli distribution by injecting stochasticity (Jang et al., 2017) into the learning of the masks Σ_{mask}:

$$\tilde{b}_t = \text{sigmoid}\left[\log\frac{\hat{b}_t u}{(1-\hat{b}_t)(1-u)}\right]^{1/\tau} \quad (5)$$

$$u \sim \text{Uniform}(0,1) \quad (6)$$

At the start of training, $W_{\text{learnable}}$ is initialised such that the mask selects all singular values and through training, its parameters learn sparser masks to achieve the target compression rate. During training, the only learnable parameters in the model are the newly introduced masking layers and the rest of the model is frozen, keeping the training process efficient.

4.3. Training

4.3.1. Distillation Dataset

On the training corpus, a set of 3000 documents, we create a distillation dataset which contains the hidden state of each token from the middle layer and the hidden states from before the logits layer (Wang et al., 2020a). The activations in this dataset will serve as labels used during training.

4.3.2. Optimisation

The objective function focuses on minimising the number of singular values selected, minimising the divergence between the activations of the original and compressed models, and enforcing the smoothness of the learned masks. We aim to minimise the total loss L, which is a weighted sum of the compression loss $L_{\text{compression}}$, the distillation loss $L_{\text{distillation}}$, and the Total Variation loss $\mathcal{L}_{\text{tv}}$:

$$\mathcal{L} = \alpha\mathcal{L}_{\text{distillation}} + \beta\mathcal{L}_{\text{compression}} + \gamma\mathcal{L}_{\text{tv}}, \quad (7)$$

where $\alpha, \beta, \gamma \in \mathbb{R}$ are hyper-parameters. These parameters are selected to guide compression rates. This is further discussed in Section 5.4.

Compression Loss For the compression loss, we directly minimise the mean of the learnable weights, helping generate sparser masks and increasing the compression rate.

$$\mathcal{L}_{\text{compression}} = \frac{1}{N_{\text{layers}}}\sum_{i=1}^{N_{\text{layers}}} \text{Average}(W_{\text{learnable},i}) \quad (8)$$

Through this, the model can learn different compression rates for every layer.

Distillation Loss The distillation loss minimises the differences in activations between the original model and the model being compressed using the mean-squared error loss in Eq. (9). We use the following distillation loss:

$$\mathcal{L}_{\text{distillation}} = \|A_{\text{compressed}} - A\|_F^2, \quad (9)$$

where $A_{\text{compressed}} \in \mathbb{R}^{L\times D}$ and $A \in \mathbb{R}^{L\times D}$ denote the activations of the compressed and original models, respectively, where L is the sequence length and D is the dimension of the hidden state. Following (Wang et al., 2020a), we use the activations of the middle layer and the hidden states before the logits layer as targets.

Total Variation Loss Theoretically, reconstructing a weight matrix after SVD involves utilising the top-k largest singular values sorted by their magnitude. Motivated by this, we introduce a Total Variation (TV) loss on the learnable mask from Eq. (4) to encourage *smooth* masks — ones that select contiguous singular values rather than skipping intermediate ones — which better aligns the learned selection with this theoretical prior.

$$\mathcal{L}_{\text{tv}} = \sum_{n=0}^{N-1} \left|\Sigma_{\text{mask},n+1} - \Sigma_{\text{mask},n}\right| \quad (10)$$

Here, n refers to the index of the mask vector of length N. For example, if the 5th and 7th singular values are selected by the mask, this loss penalises skipping the 6th, encouraging it to also be selected. We experimentally analyse the contribution of the TV loss in our ablations in Section 7.2.

4.4. Model Post-Processing

After training, we use the learned singular value masks to select the required singular values for compression. To obtain the binary mask, we applied a 0.5 threshold to the sigmoid of the logits, without needing the stochastic masking operator.

If a layer is negligibly compressed (less than 1% compressed), we reconstruct its entire weight matrix using its full rank, and the layer remains uncompressed. This design choice of avoiding using the mask is crucial to maintain performance; even if the learned singular value mask retains 90% of the singular values, compression might not be achieved as per Eq. (2), and using the full rank will better preserve performance. Empirically, we find this heuristic highly effective for performance and is explored in Ablation Section 7.3.

After training, for compressed weights, we retain only the selected left and right singular vectors. The corresponding singular values in Σ_r are absorbed by multiplying them into U_r, yielding a single matrix product $(U_r\Sigma_r)V_r^T$ and avoiding the need to store Σ_r explicitly, reducing both storage and inference overhead.

5. Experimental Setup

5.1. Evaluation

We evaluate the compressed model on zero-shot PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), OpenbookQA (Mihaylov et al., 2018), MMLU (Hendrycks et al., 2021) and 5-shot evaluations on NQ-Open (Kwiatkowski et al., 2019). These are English datasets that cover commonsense reasoning and open-domain QA.

5.2. Training and Data Configuration

The training dataset for LLRC consisted of 3,000 unique documents from WikiText-2 (Merity et al., 2016). To have a greater number of token activations present per batch, all documents selected had more than 150 words. Experiments were conducted using a batch size of 4, a maximum token length of 256, and optimised with the AdamW optimiser (Loshchilov and Hutter, 2019). To avoid unnecessary training, we use early stopping, terminating the training 750 steps after the target parameter ratio is achieved.

5.3. Masking Layer Initialization

The learnable weight matrix $W_{\text{learnable}}$ is initialised such that the model starts in an uncompressed state, selecting all singular values. Given that larger singular values are theoretically more significant, we introduce an inductive bias into the weight initialisation of the mask. To this end, $W_{\text{learnable}}$ is initialised with linearly distributed values ranging in $[3, 6]$. This range helps start training in a fully uncompressed state, while the linear spread encodes a mild inductive bias favouring the retention of higher-indexed singular values. Following (Nawrot et al., 2024), we use a temperature of 0.1 for the Gumbel function.

5.4. Objective Function Configuration

The objective function in Eq. (7) represents a weighted sum of the compression, distillation, and total variation loss. The total variation loss weight, γ, is a constant 1. The weight on the compression loss, denoted by β, is set to a constant value of 1 until the target compression ratio is achieved, after which it is set to 0 to prevent further unnecessary compression and focus on the model loss. Instead of using a constant value for α, which is more susceptible to the initial choice, inspired by (Fu et al., 2019), we oscillate α between two bounds, 1 and 0, using the cosine function. This also enables the training to oscillate focus between optimising for compression and performance.

6. Results

6.1. Evaluation Benchmarks

To evaluate the efficacy of our gradient-based rank selection procedure, we benchmark it against a baseline rank selection, Sensitivity-based Truncation Rank Searching (STRS; Yuan et al., 2024), and Adaptive Rank Selection (ARS; Gao et al., 2024). We perform extensive compression performance comparisons on four architectures of different sizes: Llama-2-7B, Llama-3-8B, Gemma-7B, and Llama-2-13B. As a baseline algorithm, 'Fixed Rate', the rank is selected to compress each layer equally to the target parameter ratio. To compare fine-tuning free-rank selection approaches, for ARS, we only use the rank selection portion of the algorithm without the post-training fine-tuning. We performed ARS using λ=16 and γ=1 (Gao et al., 2024) on the same dataset we used. Further details of hyperparameters of these approaches can be found in Appendix Section B.

As shown in Fig. 2, our approach consistently achieves higher accuracy across most datasets, particularly at lower compression ratios. At a parameter ratio of 0.80, our method outperforms STRS on Llama-2-7B by 4.3% (NQopen), 3.9% (PIQA), and 4.6% (OpenbookQA). On Llama-2-13B, improvements reach 12% on MMLU, 3.5% on BoolQ, and 4.4% on OpenbookQA.

The competitive performance of our approach is rooted in several ablation studies on how to handle

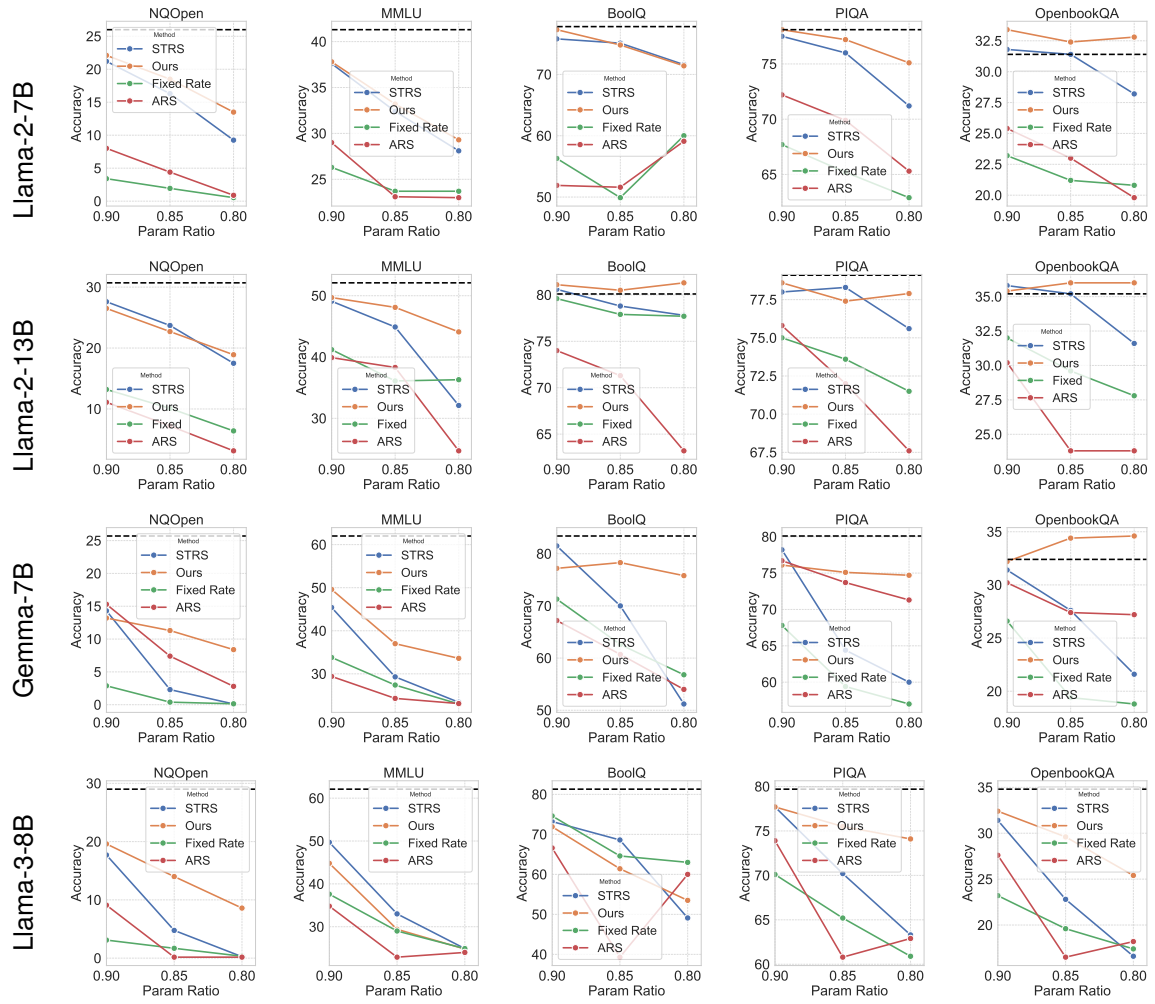

Figure 2: Model performance using different rank selection methods: Fixed Rate (naive baseline), STRS (Yuan et al., 2024), ARS (Gao et al., 2024), and our proposed approach.

Method	Param Ratio	Llama-2-7b					Llama-2-13b				
		NQ-Open	MMLU	BoolQ	PIQA	OQA	NQ-Open	MMLU	BoolQ	PIQA	OQA
Baseline	1.00	26.0	41.3	77.8	78.1	31.4	30.7	52.1	80.1	79.1	35.2
Pruner	0.90	15.5	28.5	64.8	77.3	31.0	21.2	45.4	73.4	79.4	35.2
SVD-LLM (w)	0.90	10.8	24.0	49.0	69.4	28.2	18.3	31.6	76.6	73.9	30.6
Ours	0.90	22.1	37.8	77.3	78.1	33.4	26.5	49.7	81.1	78.6	35.4
Pruner	0.80	7.8	25.1	64.6	76.2	29.0	12.6	23.2	67.2	78.3	32.2
SVD-LLM (w)	0.80	5.87	23.8	44.4	66.1	25.6	12.9	29.6	73.9	69.8	28.8
Ours	0.80	13.5	29.3	71.4	75.1	32.8	18.9	44.1	81.3	77.9	36.0
Methods with additional fine-tuning on Alpaca dataset											
Pruner+Finetune	0.90	17.9	34.0	71.4	78.1	33.0	22.1	48.0	76.4	79.7	36.6
SVD-LLM+Finetune	0.90	14.9	33.0	67.6	75.3	30.6	20.7	45.1	80.0	77.6	33.8
Pruner+Finetune	0.80	11.1	26.7	67.8	77.8	30.4	16.2	32.1	72.9	79.2	35.4
SVD-LLM+Finetune	0.80	11.8	27.1	67.5	72.4	29.6	17.6	41.3	78.6	76.3	32.4

Table 1: Performance comparison between LLM-Pruner (Ma et al., 2023), SVD-LLM (Wang et al., 2025), and our approach on LLAMA-2-7B and LLAMA-2-13B.

learning singular value masks. For instance, as described in Ablations Section 7.3, we find that our post-training heuristic of ignoring the learned masks for trivially compressed layers is effective

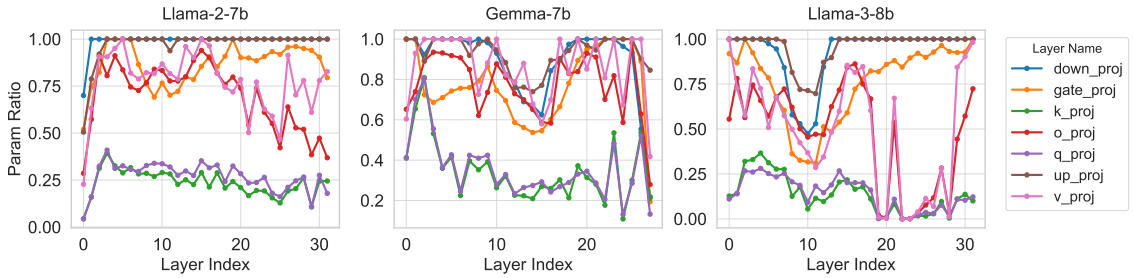

Figure 3: Distribution of learned compression rates across layer types and layer numbers in models that were compressed by 20% using our approach.

in retaining performance. The numerical results corresponding to Fig. 2 are tabulated in Appendix Section F.

6.2. Impact of Fine-Tuning

To place our work in the broader context of model compression, we benchmark our approach against LLM-Pruner and SVD-LLM. LLM-Pruner (Ma et al., 2023) is a structured pruning method that supports both fine-tuning and fine-tuning-free variants. SVD-LLM is a truncation-aware low-rank compression method, which also relies on post-compression fine-tuning for improved performance recovery (Wang et al., 2025). Another relevant method is A3 (Wong et al., 2025); however, we could not include it in our benchmarks as it was recently released and lacks a public implementation.

We conduct experiments on Llama-2-7B, Llama-2-13B, and Llama-3-8B. Because the publicly available SVD-LLM implementation does not support Llama-3-8B, our comparisons with LLM-Pruner on this model is present in Appendix Section E. Since our approach does not update model weights, we focus primarily on comparing against the fine-tuning-free variants of SVD-LLM and LLM-Pruner. Nevertheless, we also report results for their fine-tuning variants and observe that our method remains competitive even when those methods leverage additional fine-tuning. The hyperparameters used in these experiments can be found in Appendix Section B.

The comparative analysis in Table 1 demonstrates that our method significantly outperforms the fine-tuning-free variants of LLM-Pruner and SVD-LLM on both Llama-2-7B and Llama-2-13B. For example, at a parameter ratio of 0.80 on Llama-2-13B, our approach surpasses SVD-LLM (w) on all evaluated datasets and outperforms LLM-Pruner on 4 out of 5 datasets. This demonstrates that our rank selection approach better preserves performance in a fine-tuning-free manner.

In the fine-tuning setting, our approach—despite requiring no fine-tuning—consistently outperforms SVD-LLM even when SVD-LLM is fine-tuned, and remains competitive with the fine-tuned variant of LLM-Pruner. At a parameter ratio of 0.80 on Llama-2-13B, our method exceeds the performance of fine-tuned LLM-Pruner on 4 out of 5 datasets. However, on Llama-3-8B, LLM-Pruner with fine-tuning achieves higher accuracy than our method, as shown in Table 2.

6.3. Insights into Learnt Compression Rates

To understand the robustness of our rank selection method, we analyse how our training procedure discovers optimal compression strategies. To do this, we examine the learned singular value masks across different models after training. As visualised in Fig. 3, the learned compression rates demonstrate that optimisation captures the varying compressibility of layers: earlier layers, along with key and query projection layers, consistently exhibit the highest compression rates across all models, while feedforward layers (up/down projections) remain largely uncompressed.

7. Ablations

7.1. Selecting Any-k Singular Values vs Top-k

Theoretically, when selecting k singular values for retention, the optimal choice to minimise reconstruction loss would typically be the top k singular values by magnitude. However, our approach, which allows the mask to learn to select any k singular values rather than strictly the top k, has shown strong practical performance despite limited theoretical backing.

We compressed the LLaMA-2-7b model and performed evaluation using both the top-k and any-k singular value selection. Results in Figure 4 demonstrate that in the lowest parameter ratio of 0.80, any-k mask outperforms the top-k mask on all datasets. Significant gains are seen in open-domain QA with any-k performing 4% better at parameter ratio 0.80.

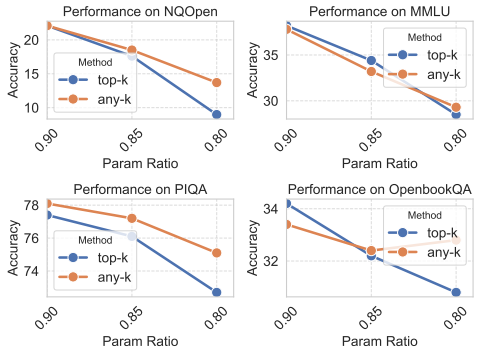

Figure 4: Evaluation performance of Llama-2-7b with using any-k and top-k masking

7.2. Introduction of Total Variation Loss

Motivated by the theoretical understanding that higher singular values are more relevant, we used the Total Variation (TV) loss to guide our learned masks to be smooth.

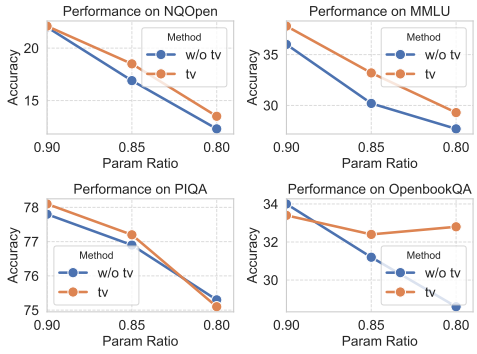

Figure 5: Evaluation performance of Llama-2-7b with and without using total variational loss

To validate this hypothesis, we compressed the LLaMA-2-7b model to various target parameter ratios, using and without using the total variational loss. Results in Figure 5 demonstrate the efficacy of introducing the total variational loss. On NQ-Open, MMLU and PIQA introducing this loss function consistently leads to higher performance. On OpenbookQA, this loss function leads to higher performance on parameter ratios of 0.85 and 0.80.

7.3. Post-Training Model Conversion

As outlined in Section 4.4, we finalise model compression by retaining only the singular vectors selected by the mask. However, as implied by Eq. (2), even if the learnt layer drops singular values, it does not always lead to compression. In cases where layers remain effectively uncompressed, we find it highly effective to disregard the learned masks for these layers and instead reconstruct the original matrix as a standard linear layer. This approach is not only theoretically justified but also empirically validated, as illustrated in Fig. 6.

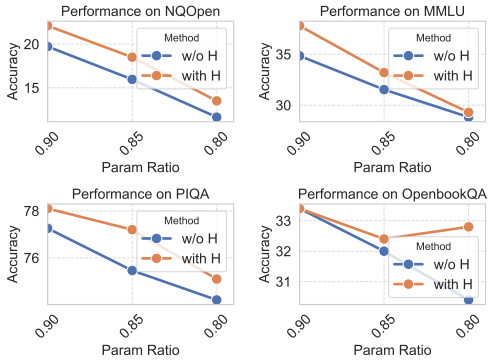

Figure 6: Performance comparison with and without our heuristic for compression: Llama-2-7b

By applying this heuristic, our method consistently outperforms baseline approaches on nearly all downstream tasks. To the best of our knowledge, this has been unexplored in prior gradient-based rank selection compression methods.

8. Conclusion

We introduced Learning to Low-Rank Compress (LLRC), a gradient-based compression method that learns optimal SVD ranks from a small calibration set. By casting rank selection as a learnable mask vector, LLRC trains more efficiently than search procedures that rely on recurrent neural networks (e.g., ARS) and, unlike most alternatives, retains *any-k* rather than just the top-k singular values. Empirically, LLRC consistently outperforms all prior rank-selection approaches across Llama-2-7B/13B, Llama-3-8B, and Gemma-7B; at 20% compression on Llama-2-13B, for example, it improves over STRS by 12% on MMLU and by 4% on OpenbookQA. Compared to existing compression techniques, our fine-tuning-free method outperforms LLM Pruner and SVD-LLM (without fine-tuning) on various compression rates. Compared to their fine-tuning variants, it outperforms SVD-LLM and remains competitive with LLM-Pruner. Beyond raw performance, the learned parameterised masks also reveal systematic compressibility patterns, which may provide insights for general compression research.

9. Limitations

The architecture in our proposed method for compression employs a linear mask to selectively mask singular values during an SVD reconstruction. While this approach simplifies the training process, it introduces some limitations related to memory usage during training. During training, to

select singular values, the weight matrices are decomposed into their full rank, which results in a higher number of parameters in each layer compared to the original configuration, as described in Equation 2. Additionally, at higher compression rates, such as 20%, we observe more degradation. We believe this can be a limitation of current SVD approaches (ASVD, Fischer SVD) and their ability to minimise reconstruction loss of the weights with fewer singular values. Exploring the synergy between learnable rank selection and component-wise decomposition techniques, such as A3 (Wong et al., 2025), may therefore be a promising direction for future work.

10. Bibliographical References

Haoli Bai, Lu Hou, Lifeng Shang, Xin Jiang, Irwin King, and Michael R. Lyu. 2021. Towards efficient post-training quantization of pre-trained language models.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, and Raul Puri. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Zihan Chen. 2018. Singular value decomposition and its applications in image processing. In *Proceedings of the 2018 1st International Conference on Mathematics and Statistics*, ICoMS '18, page 16–22, New York, NY, USA. Association for Computing Machinery.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL-HLT (1)*, pages 2924–2936. Association for Computational Linguistics.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale.

Rahul Dey and Fathi M Salem. 2017. Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing.

Shangqian Gao, Ting Hua, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. 2024. Adaptive rank selections for low-rank approximation of language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 227–241, Mexico City, Mexico. Association for Computational Linguistics.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yen-Chang Hsu, Ting Hua, Sung-En Chang, Qiang Lou, Yilin Shen, and Hongxia Jin. 2022a. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations (ICLR)*.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Vimal Kumar, Priyam Srivastava, Ashay Dwivedi, Ishan Budhiraja, Debjani Ghosh, Vikas Goyal, and Ruchika Arora. 2023. Large-language-models (llm)-based ai chatbots: Architecture, in-depth analysis and their performance evaluation. In *International Conference on Recent Trends in Image Processing and Pattern Recognition*, pages 237–249. Springer.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 20336–20350. PMLR.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

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

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. 2024. Dynamic memory compression: Retrofitting llms for accelerated inference.

Yu Pan, Jing Xu, Maolin Wang, Jinmian Ye, Fei Wang, Kun Bai, and Zenglin Xu. 2019. Compressing recurrent neural networks with tensor ring for action recognition. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

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. 2023. Llama: Open and efficient foundation language models.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2020a. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. *CoRR*, abs/2012.15828.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2025. Svd-llm: Truncation-aware singular value decomposition for large language model compression.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020b. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J. Barezi, and Pascale Fung. 2019. On the effectiveness of low-rank matrix factorization for lstm model compression.

Jeffrey T. H. Wong, Cheng Zhang, Xinye Cao, Pedro Gimenes, George A. Constantinides, Wayne Luk, and Yiren Zhao. 2025. A3 : an analytical low-rank approximation framework for attention.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared llama: Accelerating language model pre-training via structured pruning.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. Smoothquant: Accurate and efficient post-training quantization for large language models.

Bosen Yang. 2021. Application of matrix decomposition in machine learning. In *2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*, pages 133–137. IEEE.

Yinchong Yang, Denis Krompass, and Volker Tresp. 2017. Tensor-train recurrent neural networks for video classification. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3891–3900. JMLR.org.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2024. Asvd: Activation-aware singular value decomposition for compressing large language models.

Emanuele Zangrando, Steffen Schotthöfer, Gianluca Ceruti, Jonas Kusch, and Francesco Tudisco. 2023. Rank-adaptive spectral pruning of convolutional layers during training.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *ArXiv*, abs/1710.01878.

A. Training Details

A.1. Distillation Scale Parameter

The distillation scale parameter α in Equation 7 equations below:

$$z = \cos\left(\frac{2\pi \cdot 10 \cdot \text{current\_step}}{\text{total\_steps}}\right) \tag{11}$$

$$\alpha = \min\left(\max(z, b), c\right) \tag{12}$$

This scaling α follows a cosine distribution that completes 1 cycle at 10% of total training steps, and the min value is capped at b and max c, which are decided based on the rate of compression required. The lower the value of b and c, the faster the model learns the required compression rate. For llama-2-7b, we use b=0.3 and c=1.0, for llama-3-8b we use b=0.25 and c=0.5, for Gemma-7b, we use b=0.5 and c=1.0. As a warmup, for the first 250 steps, we avoid any scaling and use $\alpha = 1.0$.

B. Evaluation Details

B.1. ASVD STRS Hyper-Parameters

The hyperparameters for STRS ASVD are below:

- α: 0.5
- Number of Calibration Samples: 32
- Max Sequence Length: 2048

B.2. Adaptive Rank Selection (ARS) Hyper-Parameters

The benchmarks of our ARS algorithm is based on our implementation, which is open-sourced here: GitHub. The dataset used is the same as one used in our work, Wikitext-2 (Merity et al., 2016)

- **Llama-2-7b:** $\lambda = 16$, $\gamma = 1$, lr = $1e{-3}$, Optimizer: Adam
- **Llama-3-8b:** $\lambda = 8$, $\gamma = 2$, lr = $1e{-3}$, Optimizer: Adam
- **Gemma-7b:** $\lambda = 8$, $\gamma = 2$, lr = $1e{-3}$, Optimizer: Adam
- **Llama-2-13b:** $\lambda = 16$, $\gamma = 1$, lr = $1e{-3}$, Optimizer: Adam

For Llama-2-7b, we used λ=16 and γ=1, instead of λ=8 and γ=2, because the former led to an acceptable NQ-Open performance at Param Ratio 0.90.

B.3. LLM Pruner Hyper-Parameters

This section outlines the hyperparameters used for benchmarking LLM-Pruner on downstream datasets with various compression ratios.

- **Pruning:**
  - Block-wise Pruning:
    - Block MLP Layer Start: 4
    - Block MLP Layer End: 30
    - Block Attention Layer Start: 4
    - Block Attention Layer End: 30
  - Pruner Type: Taylor
  - Taylor Strategy: `param_first`
- **Post-Training:**
  - Dataset: `yahma/alpaca-cleaned`
  - LoRA Rank: 8
  - Number of Epochs: 2
  - Learning Rate: 1e-4
  - Batch Size: 64

B.4. SVD-LLM Hyper Parameters

These are the hyperparameters for the **SVD-LLM whitening step (without finetuning)**:

- whitening_nsamples: 256
- dataset: wikitext2
- seed: 3
- model_seq_len: 2048

As per SVD-LLM (Wang et al., 2025), after the whitening step, LoRA fine-tuning is performed separately for the left and right singular values. This is the set of hyperparameters used for each of the two fine-tuning steps.

- data_path: yahma/alpaca-cleaned
- lora_r: 8
- num_epochs: 1
- learning_rate: 1e-4
- batch_size: 64

C. Complexity Analysis of a Low-Rank Linear Layers

C.1. Space Complexity

The compression rate of a low-rank layer is derived in Section 3.1. As shown in Equation 2, a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ that originally requires mn parameters can be represented in low-rank form using only $mr + nr$ parameters for rank r. When $r < \min(m, n)/2$, this representation yields a reduction in memory usage.

C.2. Time Complexity

When applying SVD-based compression to a weight matrix $W \in \mathbb{R}^{m \times n}$, we decompose it into two lower-rank matrices $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$, where $r < \min(m, n)$ represents the compressed rank. For efficiency, this decomposition is performed after fusing the singular value vector into the left eigenvectors.

Original FLOPs: For an uncompressed linear layer with weight matrix $W \in \mathbb{R}^{m \times n}$ and input $X \in \mathbb{R}^{m \times m}$, the matrix multiplication XW requires:

$$\text{FLOPs}_{\text{original}} = 2m^2n \tag{13}$$

Compressed FLOPs: After SVD decomposition, the computation becomes $X(UV) = (XU)V$, requiring two sequential matrix multiplications:

$$\text{FLOPS}_{\text{compressed}} = 2m^2r + 2 \cdot mrn \tag{14}$$

FLOP Ratio: The computational savings achieved through low-rank decomposition is:

$$\begin{aligned} \text{FLOP Ratio} &= \frac{\text{FLOPs}_{\text{compressed}}}{\text{FLOPs}_{\text{original}}} \\ &= \frac{2mr(m+n)}{2m^2n} = \frac{r(m+n)}{mn} \end{aligned} \tag{15}$$

If the FLOP ratio is < 1, we can say that fewer number of FLOPs is required for the compressed layer. For typical compression scenarios where $r \ll \min(m, n)$ and $m \approx n$, this ratio approaches $\frac{r}{n}$, demonstrating a reduction in FLOPs of the low-rank compressed layer.

For layers with minimal compression, for example, only 5% compression, low-rank decomposition can provide limited gains for that layer. For instance, parallelisation inefficiency from requiring two separate matrix multiplications instead of one for very similar FLOPs. It would require additional data movement between HBM and SRAM. That being said, FLOP improvements can be better realised through efficient fused kernel implementations for low-rank linear projections.

D. Implementation of ARS

Our benchmarking of ARS (Gao et al., 2024) initially used a slight modification of the original approach, in which the full hypernetwork—including the GRU module—was instantiated separately for each layer to be compressed. In response to feedback, we later aligned our implementation with the original paper by using a single GRU hypernetwork shared across all layers, with only the linear projection layers instantiated per compressed layer. This change resulted in negligible differences in performance metrics; moreover, the initial implementation yielded slightly better results. Therefore, for fairness and competitiveness, we report the best-performing configuration. The full implementation has been open-sourced here.

E. More LLM Pruner Benchmarks

In addition to the models reported in Table 1, we also evaluate performance on Llama-3-8B. Because the current open-source implementation of SVD-LLM does not support Llama-3-8B, we include another comparison between our approach and LLM-Pruner only. The results are presented in Table 2.

Method	Param Ratio	LLaMA-2-7b					LLaMA-3-8b				
		NQ-Open	MMLU	BoolQ	PIQA	OQA	NQ-Open	MMLU	BoolQ	PIQA	OQA
Baseline	1.00	26.0	41.3	77.8	78.1	31.4	29.0	62.1	81.3	79.7	34.8
Pruner	0.90	15.5	28.5	64.8	77.3	31.0	17.0	41.7	68.4	77.9	31.0
Ours	0.90	22.1	37.8	77.3	78.1	33.4	19.6	44.8	71.9	77.7	32.4
Pruner	0.85	13.1	24.8	67.9	77.5	31.0	11.8	35.3	56.6	77.4	28.8
Ours	0.85	18.5	33.2	74.8	77.2	32.4	14.0	29.4	61.4	75.5	29.6
Pruner	0.80	7.8	25.1	64.6	76.2	29.0	5.5	23.0	53.5	74.2	24.4
Ours	0.80	13.5	29.3	71.4	75.1	32.8	8.6	24.8	53.5	74.1	25.4
LLM Pruner performance with additional fine-tuning on Alpaca dataset											
Pruner+Finetune	0.90	17.9	34.0	71.4	78.1	33.0	19.0	48.0	76.1	79.5	35.0
Pruner+Finetune	0.85	14.7	31.4	72.2	78.7	33.0	14.7	42.9	74.4	79.0	30.8
Pruner+Finetune	0.80	11.1	26.7	67.8	77.8	30.4	10.6	32.9	70.2	78.2	29.8

Table 2: Performance comparison between LLM-Pruner and Our approach on LLAMA-2-7B and LLAMA-3-8B

F. Tabulated Performance of Rank-Selection Models

F.1. Performance on Llama-2-7b/Llama-3-8b

Method	Param Ratio	LLaMA-2-7b					LLaMA-3-8b				
		NQOpen	MMLU	BoolQ	PIQA	OQA	NQOpen	MMLU	BoolQ	PIQA	OQA
Baseline	1.00	26.0	41.3	77.8	78.1	31.4	29.0	62.1	81.3	79.7	34.8
Fixed Rate	0.90	3.41	26.3	56.3	67.7	23.2	3.10	37.6	74.6	70.1	23.2
STRS	0.90	21.2	37.6	75.8	77.5	31.8	17.7	49.7	73.2	77.7	31.4
ARS	0.90	8.01	29.0	51.9	72.2	25.4	9.09	34.8	66.6	73.9	27.6
Ours	0.90	22.1	37.8	77.3	78.1	33.4	19.6	44.8	71.9	77.7	32.4
Fixed Rate	0.85	1.94	23.7	49.9	65.2	21.2	1.69	29.0	64.6	65.2	19.6
STRS	0.85	16.3	32.4	75.1	76.0	31.4	4.76	33.0	68.6	70.2	22.8
ARS	0.85	4.40	23.1	51.6	69.9	23.0	0.17	22.9	39.3	60.8	16.5
Ours	0.85	18.5	33.2	74.8	77.2	32.4	14.0	29.4	61.4	75.5	29.6
Fixed Rate	0.80	0.53	23.7	60.0	62.9	20.8	0.33	24.9	63.0	60.9	17.4
STRS	0.80	9.25	28.1	71.6	71.2	28.2	0.25	24.9	49.1	63.3	16.6
ARS	0.80	0.89	23.0	59.1	65.3	19.8	0.17	24.0	60.0	62.9	18.2
Ours	0.80	13.5	29.3	71.4	75.1	32.8	8.59	24.8	53.5	74.1	25.4

Table 3: Comparison of evaluation performance using different rank selection methods on LLaMA-2-7b and LLaMA-3-8b. We compare Fixed Rate (naive baseline), STRS (Yuan et al., 2024), ARS (Gao et al., 2024), and our proposed approach.

F.2. Performance on Gemma-7B

This section contains the table of metrics of Gemma-7B.

Method	**Param Ratio**	**NQ-Open**	**MMLU**	**BoolQ**	**PIQA**	**OpenbookQA**
Original	1.00	25.7	62.0	83.4	80.1	32.4
Fixed Rate	0.90	2.88	33.8	71.3	67.8	26.6
STRS	0.90	14.3	45.4	81.5	78.2	31.4
ARS	0.90	15.3	29.4	67.2	76.7	30.2
Ours	0.90	13.2	49.6	77.2	76.1	32.2
Fixed Rate	0.85	0.39	27.4	62.6	59.4	19.4
STRS	0.85	2.30	29.3	70.0	64.4	27.6
ARS	0.85	7.40	24.3	60.7	73.7	27.4
Ours	0.85	11.3	37.0	78.3	75.1	34.4
Fixed Rate	0.80	0.14	23.0	56.8	57.0	18.8
STRS	0.80	0.11	23.4	51.2	60.0	21.6
ARS	0.80	2.80	23.1	54.0	71.3	27.2
Ours	0.80	8.39	33.6	75.8	74.7	34.6

Table 4: Evaluation performance of Gemma-8B using different rank selection methods

F.3. Performance on Llama-2-13B

Method	Param Ratio	NQOpen	MMLU	BoolQ	PIQA	OpenbookQA
Original	1.00	30.7	52.1	80.1	79.1	35.2
Fixed	0.90	13.2	41.2	79.6	75.0	32.0
STRS	0.90	<u>27.6</u>	49.1	80.6	78.0	<u>35.8</u>
ARS	0.90	11.1	39.9	74.0	75.8	30.2
Ours	0.90	26.5	<u>49.7</u>	<u>81.1</u>	<u>78.6</u>	35.4
Fixed	0.85	10.1	36.1	77.9	73.6	29.6
STRS	0.85	<u>23.7</u>	44.9	78.8	<u>78.3</u>	35.2
ARS	0.85	7.22	38.3	71.3	72.0	23.8
Ours	0.85	22.7	<u>48.1</u>	<u>80.5</u>	77.4	<u>36.0</u>
Fixed	0.80	6.4	36.3	77.7	71.5	27.8
STRS	0.80	17.5	32.1	77.8	75.6	31.6
ARS	0.80	3.13	24.7	63.2	67.6	23.8
Ours	0.80	<u>18.9</u>	<u>44.1</u>	<u>81.3</u>	<u>77.9</u>	<u>36.0</u>

Table 5: Evaluation performance of Llama-2-13B using different rank selection methods

G. More Ablations

G.1. Ablation: Pre-Training Objective

G.1.1. Pre-Training Objective Ablation

In our final results, we utilised a distillation objective that minimised the divergence between the activations of the compressed model and the original model. While both methods lead to very similar results, as shown in Fig. 7, indicating the flexibility of our masking training procedure, learning the optimal ranks through distillation leads to better generalisation on the majority of datasets. At 20% compression, distillation outperforms pre-training in all datasets.

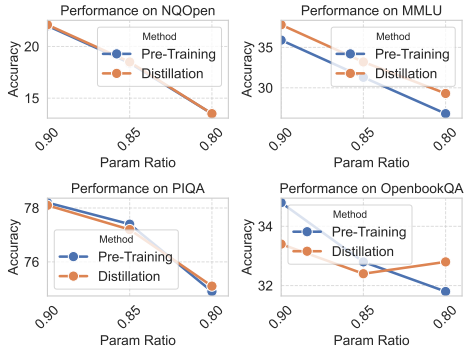

Figure 7: Comparison of performance of models trained using distillation objective and a pre-training objective (next word prediction)

For the pre-training objective, the same dataset was used and documents were packed to a sequence length of 256 tokens.

G.1.2. Pre-Training Objective Hyper-Parameters

The hyperparameters used in the training process for the pre-training objective are summarised in this section. We sampled 70,000 documents from the Wikitext dataset (Merity et al., 2016), which was the same data source used for the model trained on the distillation objective. We utilised document packing, ensuring that all documents contained 256 tokens. Besides this, all parameters, such as optimiser, maximum number of tokens, learning rate, and early stopping criteria, were the same as the experiment that used distillation, as mentioned in Section 5.2.