Adaptive Feature-based Low-Rank Compression of Large Language Models via Bayesian Optimization

Anonymous ACL submission

Abstract

In recent years, large language models (LLMs) have driven advances in natural language processing. Still, their growing scale has increased the computational burden, necessitating a balance between efficiency and performance. Low-rank compression, a promising technique, reduces non-essential parameters by decomposing weight matrices into products of two low-rank matrices. Yet, its application in LLMs has not been extensively studied. The key to low-rank compression lies in low-rank factorization and low-rank dimensions allocation. To address the challenges of low-rank compression in LLMs, we conduct empirical research on the low-rank characteristics of large models. We propose a low-rank compression method suitable for LLMs. This approach involves precise estimation of feature distributions through pooled covariance matrices and a Bayesian optimization strategy for allocating low-rank dimensions. Experiments on the LLaMA-2 models demonstrate that our method outperforms existing strong structured pruning and low-rank compression techniques in maintaining model performance at the same compression ratio.[1]

1 Introduction

In recent years, the emergence and application of large language models (LLMs) have served as a powerful stimulant for natural language processing and artificial intelligence (OpenAI, 2022, 2023; Bubeck et al., 2023; Yang et al., 2023). Adhering to the scaling law (Kaplan et al., 2020; Hoffmann et al., 2022), researchers are continually seeking LLMs with more parameters and training data, aiming to achieve general models closer to human capabilities. However, larger language models imply a larger overhead of computing resources. Therefore, when deploying LLMs, it is necessary to strike a balance between efficiency and performance (Wan et al., 2024). To achieve efficient LLMs, many compression techniques for LLMs are proposed, such as pruning (Frantar and Alistarh, 2023a; Sun et al., 2024; Ma et al., 2023), quantization (Frantar et al., 2023; Lin et al., 2023; Liu et al., 2023) and knowledge distillation (Gu et al., 2024).

Among these methods, unstructured pruning and quantization can reduce the number of parameters or memory requirements by half or even more without significant performance degradation, but they require specialized GPU kernels to fully realize their acceleration potential. In contrast, structured pruning can produce lightweight models that do not rely on specialized hardware. Despite extensive research, the performance of structured pruning still lags significantly behind that of the original model. Low-rank compression (LRC) (Ben Noach and Goldberg, 2020; Li et al., 2023) is another promising compression technique. It decomposes the weight matrix into the product of two dense low-rank matrices, discarding unimportant parameter information during the decomposition process. However, LRC remains under-explored in LLMs.

The keys to LRC are low-rank decomposition methods and low-rank dimension allocation. Existing decomposition methods can generally be categorized into two types: weight-based and feature-based decomposition. The former minimizes the reconstruction error of weight matrices by applying truncated SVD or weighted SVD (Ben Noach and Goldberg, 2020; Hsu et al., 2022; Hua et al., 2022). However, recent research (Chen et al., 2021; Yu and Wu, 2023) has discovered that the weights of most Transformer-based language models are typical of high rank or even close to full rank; thus, direct decomposition might result in significant error. In contrast, the model's features usually exhibit low-rank characteristics. Thus, more work focuses on the feature-based decomposition (Chen et al., 2021; Yu and Wu, 2023; Kaushal et al., 2023), which aims to minimize the reconstruction error

[1]The implementation code and model checkpoints are available at `https://github.com/anonymous`.

of features. On the other hand, allocating suitable low-rank dimensions to different weight matrices according to the target compression ratio can also reduce the downside on the model's overall performance since they exhibit varying sensitivities to low-rank compression.

When LRC is applied to LLMs, it encounters more new challenges. First, it is challenging for LLMs to maintain their generality while achieving feature-based low-rank compression. This is because the feature space of LLMs is extremely high dimensional, making the feature distribution more complex, and the presence of outlier features may interfere with the accurate distribution estimation. Thus, we utilize the pooled covariance matrix instead of the sample covariance matrix, which enables a more accurate estimation of feature distributions (Raninen et al., 2022). Then, for low-rank dimension allocation, manual design struggles to achieve optimal results, and due to its vast search space, grid search requires a considerable amount of time. We conduct empirical studies on the low-rank sensitivity of different types of parameters and observe significant variations among them. Based on these findings, we categorize the parameters into groups, allowing each group to share the same low-rank dimensions. This approach effectively narrows down the search space, and furthermore, we utilize sample-efficient Bayesian optimization to determine the optimal low-rank allocation. To evaluate the effectiveness of our proposed LRC method, we conduct experiments on two commonly used LLaMA-2 models (Touvron et al., 2023). Experimental results demonstrate our proposed method can perform better than existing strong structured pruning and LRC methods in LLMs. When combined with efficient post-training, our method obtains the latest state-of-art for the same settings, maintaining 98% of the model's performance at the 20% compression rate.

Overall, our main contributions include:

- We analyze the challenges that LLMs face in low-rank compression and demonstrate that LLMs represented by LLaMA exhibit vastly different sensitivities to low-rank compression across various parameters through empirical research.
- We propose a novel **B**ayesian **o**ptimization-based feature **l**ow-r**a**nk **co**mpression (**Bolaco**).
- Extensive experiments show that our Bolaco outperforms the existing strong structured pruning and LRC methods in LLMs.

2 Preliminary

In this section, we summarily introduce the foundation of low-rank factorization in model compression, and then empirically show that different layers of the Transformers-based generative large language model have different low-rank sensitivities.

2.1 Weight-based and Feature-based Low-rank Decomposition

The low-rank decomposition reduces the number of parameters by decomposing the linear layer weights into two low-rank matrices. Weight-based factorization is one naive method. For a linear layer $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$, according to the Eckart–Young–Mirsky theorem, the trunced singular value decomposition (SVD) provides the optimal solution: $\boldsymbol{W} = \boldsymbol{U\Sigma V}^T, \boldsymbol{A} = \boldsymbol{V}_r^T, \boldsymbol{B} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r$, where $\boldsymbol{A} \in \mathbb{R}^{r \times d_1}, \boldsymbol{B} \in \mathbb{R}^{d_2 \times r}$, $\boldsymbol{\Sigma}_r$ is the top-r largest singular values, $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ are the corresponding singular vectors. If $r < d_1 d_2/(d_1 + d_2)$, the factorization can reduce the total parameter amount. However, in the vast majority of cases, the weights of PLMs have a high rank, and a direct truncated SVD decomposition on the weights would lead to significant reconstruction errors (Chen et al., 2021). In comparison, the representation space of PLMs exhibits a clear low-rank property (Yu and Wu, 2023). Therefore, another line of work has considered feature-based factorization:

$$\begin{aligned} &\min_{\boldsymbol{B},\boldsymbol{A}} ||\boldsymbol{WX} - \boldsymbol{BAX}||_F \\ &\text{s.t. } \mathrm{rank}(\boldsymbol{BA}) = r. \end{aligned} \tag{1}$$

For the linear layer $\boldsymbol{Y} = \boldsymbol{WX}$, Chen et al. (2021) obtain the optimal solution to Eq. 1 by simultaneously performing the SVD decomposition of the weight and features. Yu and Wu (2023) propose a more efficient Atomic Feature Mimicking (AFM) method, which utilizes the PCA decomposition to find the projection matrices:

$$\begin{aligned} Cov(\boldsymbol{Y}) &= \boldsymbol{U\Sigma U}^T \\ \boldsymbol{Y} - E[\boldsymbol{Y}] &= \boldsymbol{U}_r\boldsymbol{U}_r^T(\boldsymbol{WX} - E[\boldsymbol{Y}]), \end{aligned} \tag{2}$$

where $Cov(\boldsymbol{Y}) \in \mathbb{R}^{d_2 \times n}$, $E[\boldsymbol{Y}] \in \mathbb{R}^{d_2}$ is the covariance and mean of features. Thus, the original linear layer can be replaced by $\boldsymbol{B} = \boldsymbol{U}_r \in \mathbb{R}^{d_2 \times r}$, $\boldsymbol{A} = \boldsymbol{U}_r^T\boldsymbol{W} \in \mathbb{R}^{r \times d_1}$ and the bias compensation $\boldsymbol{b} = (\boldsymbol{I} - \boldsymbol{U}_r\boldsymbol{U}_r^T)E[\boldsymbol{Y}]$. We have observed that the current mainstream LLMs also exhibit characteristics of high-rank weights and low-rank features. Therefore, in this paper, we focus on the feature-based low-rank factorization.

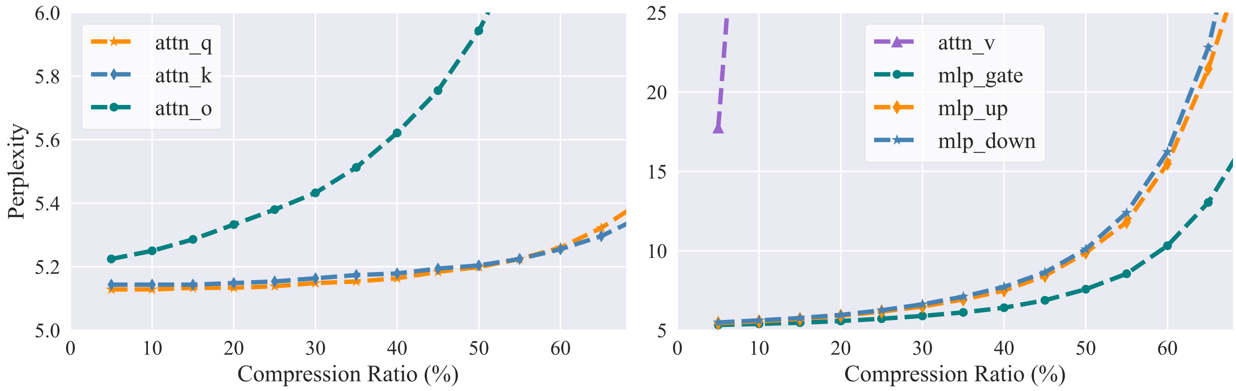


Figure 1: Sensitivity of different types of layers to low-rank compression. Each curve represents the compression of only that parameter type, with the horizontal axis indicating the compression ratio for that specific parameter type.

2.2 Different Layers Exhibit Varying Degrees of Low-rank Sensitivity

Another challenge in LRC is allocating varying low-rank compression rates to different layers. Previous works have empirically or theoretically demonstrated that different components of Transformer-based masked language models and visual models exhibit distinct low-rank properties, such as the features of the self-attention modules having a lower rank compared to those of the feed-forward modules (Dong et al., 2023; Anagnostidis et al., 2022). These findings provide prior guidance for low-rank compression. However, detailed studies on current mainstream LLMs are still lacking. Therefore, we take the LLaMA-v2-7b as an example to study the low-rank sensitivity within each layer across different types of layers and the same type of layers. Llama-family LLMs have seven distinct parameter categories: *attn_q*, *attn_k*, *attn_v*, *attn_o*, *mlp_up*, *mlp_down*, and *mlp_gate*. We evaluate the perplexity changes on Wikitext-2 (Merity et al., 2016) for each category under varying low-rank compression ratios. As Figure 1 shows, at the same low-rank compression rate, distinct layers exhibit notable performance variations. For *attn_q* and *attn_k*, they demonstrate robustness to low-rank compression, with an increase in perplexity not exceeding 2% even at a compression rate of 60%. In contrast, *attn_v*, with an equivalent parameter count, exhibits high sensitivity, leading to a significant surge in perplexity with compression rates even below 5%. Therefore, assigning the same low-rank compression rate to different types of layers during low-rank compression of LLM is a sub-optimal solution. In addition to the differences, we also observe certain similarities, e.g., *attn_q* and *attn_k* have similar low-rank sensitivities. More empirical study results are shown in Appendix A.

3 Methodology

3.1 Feature-Based Low-Rank Decomposition in High-Dimensional Spaces

An efficient feature-based low-rank decomposition method performs PCA on features to identify the optimal low-rank matrices. To achieve general task-agnostic compression, we follow the setup of prior work (Frantar and Alistarh, 2023b; Sun et al., 2023; Ma et al., 2023), utilizing a subset of the pre-training data as calibration data $\mathcal{D}_{cal} = \{x_i\}_{i=1}^{n}$. As described in Eq.2, we first estimate the covariance matrix of the entire feature space distribution $\mathcal{Y}$ with the sample covariance matrix (SCM) of the calibration data features:

$$Cov_S(\boldsymbol{Y}) = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T(\boldsymbol{y}_i - \bar{\boldsymbol{y}}), \quad (3)$$

where $\boldsymbol{y}_i$ represents the feature of x_i, $\bar{\boldsymbol{y}}$ refers to the mean of all calibration data features. However, LLMs typically have high-dimensional feature spaces (e.g., the intermediate size of LLaMA-v2-7b has exceeded 10,000 dimensions). Precisely estimating the covariance matrix in such high-dimensional spaces has always been a statistical challenge, as the SCM does not effectively estimate the covariance of high-dimensional distributions. For instance, calibration data sampled from pre-training datasets may introduce outlier features due to low-quality text or inadequate sampling. In high-dimensional spaces, these outlier features are difficult to identify due to the "curse of dimensionality", and their impact is further exacerbated in estimating high-dimensional covariance matrices due to the sparsity of data points. Thus, to estimate the covariance of the feature space more robustly and accurately, we propose using the pooled covariance matrix (PCM) in place of the SCM. We split the calibration data into m groups. For each group, we can calculate the SCM $Cov_S(\boldsymbol{Y_k})$, then the pooled covariance matrix is:

$$Cov_P(\boldsymbol{Y}) = \frac{1}{m}\sum_{k=1}^{m} Cov_S(\boldsymbol{Y}_k) \tag{4}$$

3.2 Low-Rank Allocation Based on Bayesian Optimization

As investigated in Section 2.2, different types of layers, and even each individual layer, exhibit varying sensitivities to low-rank compression. Therefore, allocating distinct compression ratios to different layers is crucial to achieve the desired compression rate with minimal performance degradation. For a LLM $f(\cdot;\boldsymbol{\theta})$, we compress it with the set of low-rank compression ratios $\boldsymbol{\lambda} = \{\lambda_i\}_{i=1}^{k}$. We use a task-agnostic evaluation dataset $\mathcal{D}$ to evaluate performance of the compressed model $f(\cdot;\boldsymbol{\theta},\boldsymbol{\lambda})$, such as the perplexity on a subset of pretraining data. Therefore, the optimization objective of low-rank allocation can be formulated as:

$$\begin{aligned} \min_{\lambda\in\mathcal{V}} H(\boldsymbol{\lambda}) &= \mathbb{E}_{(x,y)\sim\mathcal{D}} h(f(x;\boldsymbol{\theta},\boldsymbol{\lambda}), y) \\ s.t. \Sigma\boldsymbol{\lambda} &\leq \rho, \end{aligned} \tag{5}$$

where $h(\cdot,\cdot)$ is the evaluation metric, ρ is model's overall compression ratio. For LLMs, searching the optimal low-rank allocation is a challenging optimization problem. First, the impact of the low-rank count allocated to different layers on the performance of the compressed model is combinatorial, and optimizing any one component independently may lead to a locally optimal solution. Then, due to LLMs' vast number of parameters, evaluating $H(\boldsymbol{\lambda})$ is very time-consuming. Therefore, we leverage sample-efficient Bayesian optimization (BO) (Xu et al., 2022) to optimize Eq 5. BO estimates the objective $H(\boldsymbol{\lambda})$ with a stochastic surrogate model and updates the posterior estimation of $H(\boldsymbol{\lambda})$ based on the results of each search step. We utilize the Gaussian process $\mathcal{N}(\mu(\cdot),\sigma^2(\cdot))$ as the surrogate model. Given the previous $t-1$ search steps $\{\boldsymbol{\lambda}_1,\cdots,\boldsymbol{\lambda}_{t-1}\}$ and their evaluation $H_{t-1} = [H(\boldsymbol{\lambda}_1),\cdots,H(\boldsymbol{\lambda}_{t-1})]$, the surrogate model is updated as:

$$\begin{aligned} \mu(\boldsymbol{\lambda}) &= \boldsymbol{k}(\boldsymbol{K}+\eta^2\boldsymbol{I})^{-1}H_{t-1} \\ \sigma^2(\boldsymbol{\lambda}) &= k(\boldsymbol{\lambda},\boldsymbol{\lambda}) - \boldsymbol{k}^T(\boldsymbol{K}+\eta^2\boldsymbol{I})^{-1}\boldsymbol{k}, \end{aligned} \tag{6}$$

where $k(\cdot,\cdot)$ is a kernel function, $\boldsymbol{k} = (k(\boldsymbol{\lambda},\boldsymbol{\lambda}_i))_{i\in[t-1]}$, $\boldsymbol{K} = (k(\boldsymbol{\lambda}_i,\boldsymbol{\lambda}_j))_{i,j\in[t-1]}$, and $\eta^2\boldsymbol{I}$ is the white kernel to model observation noise.

After obtaining the posterior estimation of $H(\boldsymbol{\lambda})$ (i.e., $H(\boldsymbol{\lambda}) \sim \mathcal{N}(\mu(\boldsymbol{\lambda}),\sigma^2(\boldsymbol{\lambda}))$), BO determines the next compression rate allocation state through the acquisition function. Expected improvement (EI) is a popular and effective acquisition function:

$$\begin{aligned} \alpha(\boldsymbol{\lambda}) &= \mathbb{E}_{H(\boldsymbol{\lambda})}\left[\max\left\{0, H' - H(\boldsymbol{\lambda})\right\}\right] \\ \boldsymbol{\lambda}_t &= \operatorname*{argmax}_{\boldsymbol{\lambda}} \alpha(\boldsymbol{\lambda}), \end{aligned} \tag{7}$$

where $H' = \min_{i\in[t-1]} H(\boldsymbol{\lambda}_i)$, it means the minimal value observed so far. Then, BO chooses the point with the greatest EI to explore. After obtaining the optimal ratio $\boldsymbol{\lambda}^*$, we can determine the allocated rank: $r_i = (1-\lambda_i)d_1d_2/(d_1+d_2)$. To fully leverage the acceleration effect of GPU matrix multiplication, we adhere to Nvidia's user guidelines[2] by rounding the low-rank dimensions to the nearest multiple of eight.

The evaluation metric and validation data play a significant role in the optimization performance of BO. They must meet two criteria: cost-effectiveness and accurately reflect actual changes in performance. To this end, we propose a sensitive-based sampling method. This method randomly samples n allocation schemes, calculates the variance of the perplexity of each sample under different allocations, and selects the top-k samples as validation data. In addition, considering the smaller validation set may not comprehensively reflect the LLM's performance, potentially leading to over-fitting in the validation set. To prevent BO from blindly improving the compressed model's language modeling performance on the validation set, we aim to make the compressed model have a prediction distribution for the next word close to the original model. Hence, we employ the reverse KL divergence to quantify the difference:

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\lambda}) = D_{KL}(f(x;\theta)||f(x;\theta,\lambda)). \tag{8}$$

3.3 Post-training

After low-rank compression, there remains a noticeable performance gap between the compressed model and the original LLM. To further bridge this gap, following Ma et al. (2023), we perform efficient low-rank subspace post-training on the compressed model. However, if we apply the original

[2]https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html#gpu-imple

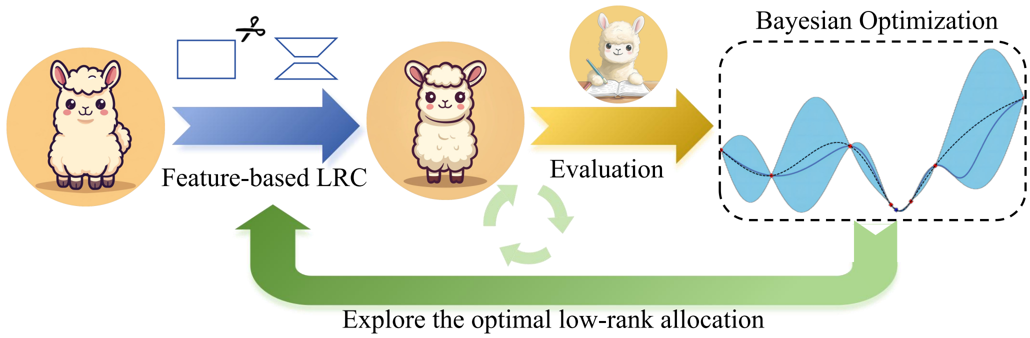


Figure 2: Illustration of our Bolaco. It initializes a low-rank dimension allocation and compresses the model via feature-based low-rank compression. Then, it evaluates the compression performance and optimizes the low-rank dimension allocation through Gaussian process-based Bayesian optimization.

LoRA (Hu et al., 2022) to the low-rank compressed model, the tunable low-rank parameters may not be in the same subspace as the low-rank compressed model parameters, leading to an increase of the parameters' rank after merging. Therefore, inspired by the ELoRA (Kopiczko et al., 2024), we select the subspace of compressed model parameters as fixed low-rank matrices and adjust the subspace by trainable vectors:

$$\boldsymbol{Y} = (\boldsymbol{B}\boldsymbol{A} + \boldsymbol{\Lambda}_b \boldsymbol{B}_{r'} \boldsymbol{\Lambda}_d \boldsymbol{A}_{r'})\boldsymbol{X}, \quad (9)$$

where $\boldsymbol{B}_{r'} \in \mathbb{R}^{d_2 \times r'}$ and $\boldsymbol{A}_{r'} \in \mathbb{R}^{r' \times d_1}$ are fixed subspace of $\boldsymbol{B}$ and $\boldsymbol{A}$, $\boldsymbol{\Lambda}_b$ and $\boldsymbol{\Lambda}_d$ are diagonal matrices. During the post-training, we only tune elements on the diagonal of $\boldsymbol{\Lambda}_b$ and $\boldsymbol{\Lambda}_d$.

4 Experiments

4.1 Baseline and Datasets

We compare our method with the competitive structured pruning and low-rank compression methods in LLMs: LLM-Pruner (Ma et al., 2023), FLAP (An et al., 2023), SliceGPT (Ashkboos et al., 2024), LoRD (Kaushal et al., 2023), ASVD (Yuan et al., 2023). We provide the detailed description of baseline methods in Appendix B.

To evaluate the effectiveness of our proposed low-rank compression method in the task-agnostic setting, we conduct experiments in seven zero-shot common sense reasoning datasets: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy/challenge (Clark et al., 2018) and OpenbookQA (Mihaylov et al., 2018). We also report the perplexity of the compressed model on the WikiText2 (Merity et al., 2016), PTB (Marcus et al., 1993), and C4 (Raffel et al., 2020) datasets to evaluate its language modeling capabilities.

4.2 Experimental Details

In our main experiments, we apply our method to LLaMA-v2-7b and LLaMA-v2-13b. We randomly select 1,024 samples from the training set of C4 as the calibration data. Each sample has a sequence length of 4,096. To estimate the covariance matrix while saving memory usage, we employ the Welford's online algorithm (Welford, 1962). For the pooled covariance matrix, we partition the calibrated data into 32 groups. During the Bayesian optimization, we utilize the Matern kernel as the covariance function.We randomly sample 20 low-rank allocation schemes and select the top-100 samples with greatest perplexity variance of Wikipedia as the evaluation data. Each sample has a sequence length of 4,096 (4k tokens). Considering that Bayesian optimization is not well-suited for high-dimensional scenarios, we conduct experiments with two settings based on the observations in Section 2.2: (a) 5×1: We allow $attn\_q$ and $attn\_k$ to share a low-rank dimension, and the same type of parameters across different layers to also share a low-rank dimension, thus BO only optimizes 5 parameters; (b) 5×4: Building on the setup of (a), we divide the model's layers into 4 groups in sequence, with no parameter sharing between different groups, resulting in BO needing to optimize 20 parameters. Moreover, given that the parameters of the FFN module are more sensitive to low-rank compression than those of the attention module, we set the length scale for the attention and FFN parameters in the Matern kernel to 1.0 and 0.8, respectively, to emphasize the more significant impact of FFN parameters' rank changes on model performance. We run 50 epochs BO to search the optimal low-rank allocation. At the post-training stage, following LLM-Pruner, we use the Alpaca

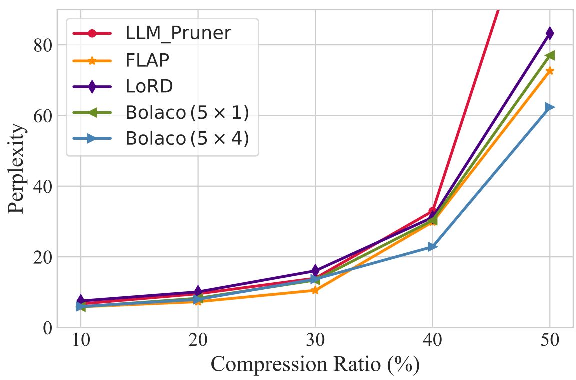


Figure 3: The perplexity of WikiText2 on LLaMA 2-7b with different compression ratios.

dataset (Taori et al., 2023) and train 2 epochs. More details can be found in the Appendix C.

4.3 Main Results

We report the perplexity of language modeling for various compression methods at different compression ratios in Figure 3 and 6, and the zero-shot common sense reasoning results in Table 1 and 5 (in Appendix). In terms of language modeling capabilities, FLAP demonstrates strong competitiveness, particularly when the compression rate exceeds 30%, where FLAP's perplexity is slightly better than our Bolaco (5×1). However, in the 7b model, Bolaco (5×4) achieves the best language modeling performance at high compression rates. Nevertheless, in the 13b model, despite Bolaco (5×4) still leading other compression techniques, it maintains a certain gap from FLAP. For zero-shot tasks, our method significantly outperforms all baselines without any further post-training, achieving an average performance increase of 1.5-2% across seven datasets. After post-training with only about 1‰ parameters and 3 hours, our method further narrows down the performance difference between the compressed model and the original model. It retains 96%-98% of the original model's performance at the 20% compression ratio, and at a 30% compression ratio, it maintains 91%-95% of the performance. Comparing the 5×1 and 5×4 setting, we find that the performance difference between the two is not significant. At the 20% compression ratio, simply allocating different low-rank dimensions to different types of parameters suffices to achieve the best current performance. However, at the 30% compression rate, the 5×4 setting outperforms the 5×1, indicating that more granular low-rank assignments contribute to enhanced performance in compressed models at higher compression rates.

5 Analysis and Discussion

5.1 Impact of Calibration Data and Covariance Estimation

Accurate estimation of feature distribution is crucial for the feature-based low-rank decomposition, which primarily depends on the number of calibration samples and the accuracy of the covariance matrix estimation. Thus, we investigate the impact of the two factors on LLaMA-v2-7b at the 20% compression ratio. In this experiment, we do not account for the effects of low-rank dimensions allocation, and maintain consistency with the settings of LoRD. As results shown in Table 2, as the calibration dataset size gradually increases, we observe a consistent improvement in both the language modeling capabilities and the performance on downstream tasks of the compressed model. Therefore, given sufficient data and computational resources, expanding the calibration dataset is a reliable method for enhancing the performance of compressed models. On the other hand, comparing the two covariance estimation methods, there is no significant difference in their language modeling capabilities. However, for downstream common sense reasoning tasks, the pooled SCM achieves an average improvement of 0.3 points across seven datasets without any additional burden.

5.2 Impact of Objective Function

We explore the impact of the objective function in the Bayesian optimization stage. We conduct experiments on LLaMA-v2-7b and report results in Table 3. Overall, incorporating the reverse KL divergence (RKL) between the compressed model and the original model's predictive distribution into the objective function can lead to a better low-rank dimensions allocation. Especially in the 5×4 setting, which is more difficult to optimize for Bayesian optimization, the performance gains from RKL term are even more obvious. We suppose that the RKL term may serve two roles. Firstly, as a regularization term, it prevents overfitting on smaller validation sets during BO. Although the compressed model exhibits a slight increase in perplexity on the language modeling dataset at the 20% compression rate with the 5×4 setting, there is a significant improvement in performance on downstream tasks. Secondly, incorporating the RKL term may

Ratio	Methods	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
0%	LLaMA-v2-7b	77.74	78.07	75.97	68.98	76.30	46.33	44.20	66.80
20%	LLM-Pruner	63.27	76.12	67.93	64.80	68.73	38.65	40.00	59.93
	LLM-Pruner (w/ PT)	66.45	76.28	70.90	65.75	70.62	39.59	43.20	61.83
	FLAP	70.21	75.24	69.34	66.30	67.30	39.42	37.40	60.74
	SliceGPT	46.73	69.04	58.98	64.33	60.31	35.07	40.40	53.55
	LoRD	72.60	73.56	63.70	65.90	69.70	37.71	39.20	60.34
	ASVD	73.61	71.93	66.05	64.17	65.24	36.26	37.40	59.24
	Bolaco (5×1)	72.17	75.52	66.76	<u>67.72</u>	<u>73.02</u>	38.74	40.60	62.08
	Bolaco (5×1 w/ PT)	73.79	**77.53**	**72.72**	**68.11**	**73.19**	**42.24**	<u>43.60</u>	**64.45**
	Bolaco (5×4)	<u>75.05</u>	75.46	67.12	67.01	72.05	38.91	42.40	62.57
	Bolaco (5×4 w/ PT)	**75.84**	<u>76.61</u>	<u>71.70</u>	65.67	72.60	<u>41.81</u>	**45.00**	<u>64.18</u>
30%	LLM-Pruner	52.51	71.93	59.49	58.72	61.41	33.96	36.60	53.52
	LLM-Pruner (w/ PT)	63.30	**76.01**	65.23	64.25	66.62	37.20	40.20	58.97
	FLAP	66.88	72.74	63.80	64.01	60.65	34.47	36.40	56.99
	SliceGPT	39.11	63.38	49.16	62.47	55.72	31.48	32.80	47.73
	LoRD	69.63	70.46	55.87	64.17	63.80	32.59	35.00	55.93
	ASVD	59.42	55.93	35.05	52.25	34.30	26.45	26.60	41.43
	Bolaco (5×1)	68.26	72.09	57.46	**65.87**	65.19	32.85	37.20	56.99
	Bolaco (5×1 w/ PT)	70.34	74.32	<u>67.81</u>	65.04	<u>69.02</u>	<u>38.31</u>	<u>41.80</u>	<u>60.95</u>
	Bolaco (5×4)	<u>70.37</u>	71.44	59.62	64.80	66.46	34.39	38.60	57.95
	Bolaco (5×4 w/ PT)	**71.83**	<u>75.19</u>	**68.03**	<u>65.67</u>	**69.15**	**38.74**	**42.40**	**61.57**

Table 1: Zero-shot performance of the compressed LLaMA-v2-7b models. w/ PT means the method with post-training. **Bold** denotes the best result at the same compression ratio, while <u>underline</u> indicates the second best result.

	Wikitext (↓)	PTB (↓)	C4 (↓)	ZS (↑)
Covariance estimate				
Naive SCM	9.96	54.69	11.46	60.34
Pooled SCM	9.93	54.68	11.45	60.64
# Samples				
128	10.55	56.29	11.99	60.26
256	10.24	55.42	11.88	60.16
512	10.30	55.03	11.61	60.56
1,024	9.93	54.68	11.45	60.64

Table 2: Impact of different covariance estimation methods and the number of calibration data. "ZS" denotes the average performance on seven zero-shot common sense reasoning datasets.

	Wikitext (↓)	PTB (↓)	Zero-shot (↑)
20%			
PPL (5×1)	8.36	48.42	61.70
w/ RKL (5×1)	8.27	47.06	62.08
PPL (5×4)	8.07	47.96	60.98
w/ RKL (5×4)	**7.96**	**45.84**	**62.57**
30%			
PPL (5×1)	13.78	71.50	56.97
w/ RKL (5×1)	13.41	70.52	56.99
PPL (5×4)	**12.65**	**68.85**	57.57
w/ RKL (5×4)	13.70	72.14	**57.95**

Table 3: Results under different objective function.

smooth the objective function, enabling the Gaussian process surrogate model to more accurately approximate the real black-box objective function.

5.3 The Transferability of Rank Allocation

In practical applications, we may utilize a variety of fine-tuned models based on the LLaMA foundation model. If we perform Bayesian optimization from scratch to optimize the low-rank allocation for each model, it will waste a significant amount of time and computational resources. Hence, we investigate whether the low-rank allocation of the base model can be transferred to the corresponding fine-tuned models. We transfer the allocation of LLaMA-v2-7b/13b to LLaMA-v2-7b/13b-chat, respectively. We consider two migration strategies: **a)** directly reusing the low-rank allocation of the base model and **b)** using the low-rank allocation of the base model as the initial value for Bayesian optimization and then optimizing only 20 epochs. As Figure 4 shows, direct reusing can achieve results that outperform all baseline methods, even the Bayesian optimization from scratch. If 20 epochs of Bayesian optimization follow reuse, there is a chance to find an even better low-rank allocation.

5.4 The Effectiveness of Validation Data Sampling

Table 4 shows results on LLaMA-v2-7b at 20% compression ratios under Wikipedia and its sampled data. The top-100 and bottom-100 represent the 100 samples with the highest and lowest per-

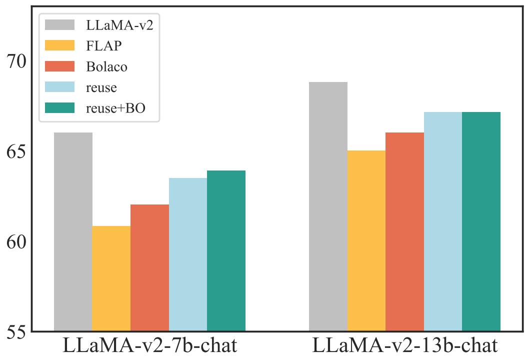


Figure 4: The average performance on zero-shot tasks about the transferability of rank allocation.

plexity variances, respectively. BO can optimize a good result when using Wikipedia and the top-100 sampled data for validation, showing that our method can sample a smaller subset for improving validation efficiency while maintaining performance comparable to the entire dataset. Conversely, with the bottom-100 sampled data, BO's optimization performance is significantly inferior, with performance similar to unoptimized LoRD. Furthermore, we observe that the top-100 samples (6.09 ppl) have higher perplexity than the bottom-100 samples (3.25 ppl) in the original LLaMA-v2-7b, indicating that the bottom-100 samples are already well-modeled and are very robust to model compression. This data may not truly reflect the performance change caused by model compression. Therefore, we suggest selecting validation data that is more sensitive to compression, typically samples with slightly worse language modeling performance. Similar to the "buckets effect", these samples may represent the performance boundaries of LLMs. Considering the performance of these samples in the optimization process can maximize the overall performance of the compressed model.

6 Related work

A common technique for low-rank factorization is SVD, which retains only the top-r largest singular values and their corresponding singular vectors to obtain two rank-r matrices. Ben Noach and Goldberg (2020) first combine SVD with knowledge distillation, applying it to compress BERT. Directly applying SVD decomposition implies an assumption that each parameter in the weight matrix equally affects the model performance. This contradicts many previous research, therefore, FWSVD (Hsu et al., 2022) and TFWSVD (Hua et al., 2022) consider weighting the weight matrix using Fisher

	Wikitext (↓)	PTB (↓)	Zero-shot (↑)
Wikipedia	7.98	46.85	62.27
Top-100	**7.96**	**45.84**	**62.57**
Bottom-100	8.38	50.41	60.90

Table 4: Results under different validation data.

information. Chen et al. (2021) observe that PLMs' weights are not inherently low-rank matrices. Therefore, directly applying SVD will result in significant reconstruction loss. However, they find that the product of data representation and weights is low-rank. Hence, they perform a global low-rank decomposition on it. Following this observation, Yu and Wu (2023) propose the atomic feature mimicking (AFM) method to decompose the output features. Ren and Zhu (2023) also observe the high rank phenomenon of PLM weights. They utilize iterative first-order unstructured pruning to reduce the rank of the weight matrix, and then apply Fisher information-weighted SVD decomposition for low-rank compression. For LLMs, low-rank compression has not yet received the attention it deserves. LoRD (Kaushal et al., 2023) applies AFM to code LLMs, demonstrating the potential of low-rank decomposition in compressing LLM. Recently, Sharma et al. (2023) conduct an in-depth study on the weight decomposition of LLMs and discover that the low-rank components of the weights encapsulate low-frequency information. By meticulously selecting low-rank components, it is possible to eliminate interfering signals and further improve LLMs' performance. However, their research does not propose a practical low-rank compression algorithm.

7 Conclusion

In this paper, we attempt to unearth the potential of low-rank compression for lightweight universal LLMs. We thoroughly investigate the challenges of low-rank compression in LLMs and the low-rank characteristics of features within LLMs. We propose a Bayesian optimization-based feature low-rank compression to address these challenges, incorporating pooled covariance estimation and Bayesian optimization for more precise feature distribution estimation and low-rank dimension allocation, respectively. Experimental results on the LLaMA 2 model demonstrate that our method significantly outperforms existing structured pruning and other low-rank compression techniques.

Limitations

Although our proposed Bolaco has made significant progress in low-rank compression for LLMs, there are still some limitations:

- Due to computational resource constraints, we only conduct thorough experiments on two commonly used LLaMA 2 models, lacking investigation into larger models (such as LLaMA 2-70B), other architectures (such as the OPT and T5 families), and multimodal models.
- To improve the efficiency of Bayesian optimization, we reduced the parameter dimensions by sharing parameters of different types and layers using low-rank dimensions. This may limit the potential performance of the model. We plan to use more advanced methods to find better low-rank allocation while maintaining flexibility.
- Compared to state-of-the-art structured pruning, low-rank compression falls short in highly compressed language models, but exhibits better zero-shot performance on downstream tasks. These observations inspire us to investigate how to effectively combine these two approaches to capitalize on their advantages in the future.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2023. Fluctuation-based adaptive structured pruning for large language models.

Sotiris Anagnostidis, Luca Biggio, Lorenzo Noci, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. 2022. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. In *Advances in Neural Information Processing Systems*.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*.

Matan Ben Noach and Yoav Goldberg. 2020. Compressing pre-trained language models by matrix decomposition. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889, Suzhou, China. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan bras, Jianfeng Gao, and Choi Yejin. 2020. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:7432–7439.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2021. Drone: Data-aware low-rank compression for large nlp models. In *Advances in Neural Information Processing Systems*, volume 34, pages 29321–29334. Curran Associates, Inc.

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 *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)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2023. Attention is not all you need: Pure attention loses rank doubly exponentially with depth.

Elias Frantar and Dan Alistarh. 2023a. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Elias Frantar and Dan Alistarh. 2023b. SparseGPT: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Ting Hua, Yen-Chang Hsu, Felicity Wang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Numerical optimizations for weighted low-rank estimation on language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1404–1416, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Ayush Kaushal, Tejas Vaidhya, and Irina Rish. 2023. Lord: Low rank decomposition of monolingual code llms for one-shot compression.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. ELoRA: Efficient low-rank adaptation with random matrices. In *The Twelfth International Conference on Learning Representations*.

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, Xingyu Dang, Chuang Gan, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. Llm-qat: Data-free quantization aware training for large language models.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. *Open AI, blog*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

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

Elias Raninen, David E. Tyler, and Esa Ollila. 2022. Linear pooling of sample covariance matrices. *IEEE Transactions on Signal Processing*, 70:659–672.

Siyu Ren and Kenny Q. Zhu. 2023. Low-rank prune-and-factorize for language model compression.

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

Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. 2023. The truth is in there: Improving reasoning in language models with layer-selective rank reduction.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. 2024. Efficient large language models: A survey.

B. P. Welford. 1962. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

Wenjie Xu, Yuning Jiang, Emilio T. Maddalena, and Colin N. Jones. 2022. Lower bounds on the worst-case complexity of efficient global optimization.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of lmms: Preliminary explorations with gpt-4v(ision).

Hao Yu and Jianxin Wu. 2023. Compressing transformers: Features are low-rank, but weights are not! *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11007–11015.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

A More Experiments on Low-rank Sensitivity

As shown in Figure 5. We further reduce the parameters by 50% on LLaMA-v2-7b by low-rank compression for each layer and test its perplexity on WikiText2. We observe that the low-rank sensitivity varies significantly across different types of parameters. Compression of $attn\_q$ and $attn\_k$ seemingly has negligible impact on overall performance across all layers. In contrast, the upper layers of mlp are more sensitive compared to the lower and middle layers. We also conduct experiments on LLaMA-7b-chat and OPT-6.7b and find significant variability between all the different types of parameters. However, for OPT-6.7b, the differences between them are less pronounced than for LLama, especially for $attn\_v$, which does not show an explosive increase in perplexity.

B Baselines

LLM-Pruner (Ma et al., 2023) is a dependency-aware one-shot structured pruning method. It evaluates the importance of each structure through a first-order Taylor expansion and prunes the structures with the lowest scores. After pruning, it uses LoRA post-training to recover performance.

FLAP (An et al., 2023) is an one-shot retraining-free structured pruning method. It utilizes a fluctuation-based metric to measure the impact of pruning on features and employs a bias term to compensate for the pruning loss.

SliceGPT (Ashkboos et al., 2024) is a post-training sparsification method. It replaces each weight matrix with a smaller matrix, reducing the embedding dimension of the network.

LoRD (Kaushal et al., 2023) is a naive feature-based low-rank compression method for code LLMs. It does not take into account the low-rank allocation of varying parameters. We migrate it to generic LLaMA-family LLMs.

ASVD (Yuan et al., 2023) is a training-free SVD-based LLM compression method. It manages activation outliers by scaling the weight matrix based on the activation distribution.

C Implementation Details

For LoRD, due to the absence of reference settings for its application on the LLaMA, we manually search a good low-rank allocation for it. At the 20% compression ratio, we do not compress $attn\_v$, and reduce the parameter count of $attn\_q/k$ by 30%, with a 20% reduction in the remaining parameters. At the 30% ratio, we reduce the parameter count of $attn\_q/k$ by 45%, with a 30% reduction in the remaining parameters except $attn\_v$.

At the post-training stage, we only add fine-tunable low-rank matrices for the compressed parameters. We set the low-rank dimension $r' = 256$, the learning rate is 2e-3, and the batch size is 64.

D Discussion on compute intensive about Bolaco

The computational cost of our method is divided into three parts:

PCA decomposition Our method requires only one PCA decomposition of the obtained representations and truncates them according to the assigned rank during the rank allocation process to generate various compressed models. The computational cost here is the same as that of the existing low-rank decomposition method LORD.

Obtaining evaluation results In our experiments, the validation set we selected is not large, about 34k tokens, so the validation process takes less time, and the total validation time spent by llama-2-7b is about 40-45min on a 40G A100.

Bayesian Optimization The computational time for 50 epochs of Bayesian optimization is approximately 45-50 minutes, which is considered acceptable in practical applications. Compared to iterative pruning, Bayesian optimization is more memory-efficient, as it only requires the memory overhead of forward propagation without storing gradients, momentum, or other optimizer states. Furthermore, as discovered in Section 5.5, the low-rank configurations optimized on a base model can be transferred directly, or with few rounds of Bayesian optimization, to variant models with the same architecture. It implies that we can quickly obtain a well-performing, low-rank compressed model for fine-tuned LLMs on different datasets in practice.

E Statistics of the Compressed Model

We report the statistic of original and compressed models in Table 7, including the parameter count, MACs and memory requirements. Statistical evaluation is conducted using the inference mode, where the model is fed a sentence consisting of 64 tokens.

To aid subsequent researchers in reproducing our results, Table 8 provides the low-rank allocations of Bolaco. The elements of the array represent the low-rank dimensions for $attn\_q/k$, $attn\_o$,

Ratio	Methods	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
0%	LLaMA-v2-13b	80.52	79.05	79.38	72.14	79.42	49.23	45.20	69.27
20%	LLM-Pruner	66.33	78.18	74.47	64.48	72.26	45.90	44.20	63.69
	LLM-Pruner (w/ PT)	67.06	**78.94**	75.92	67.32	72.69	44.28	44.60	64.40
	FLAP	71.28	76.55	74.67	69.53	72.56	44.03	42.00	64.37
	SliceGPT	45.44	71.00	62.86	68.35	71.09	41.72	41.20	57.38
	ASVD	79.36	76.61	72.82	69.69	74.54	43.00	44.60	65.80
	LoRD	78.47	76.01	69.58	71.03	74.33	40.87	44.40	64.96
	Bolaco (5 × 1)	80.00	76.50	73.25	70.24	76.18	43.86	**45.20**	66.46
	Bolaco (5 × 1 w/ PT)	**81.22**	77.69	**76.66**	**71.59**	**77.31**	**46.93**	44.00	**67.91**
	Bolaco (5 × 4)	80.58	76.22	71.44	71.19	75.38	42.49	44.00	65.90
	Bolaco (5 × 4 w/ PT)	80.95	77.64	75.84	69.93	75.25	45.14	44.20	67.00
30%	LLM-Pruner	62.45	75.90	67.90	60.22	65.45	40.36	44.60	59.55
	LLM-Pruner (w/ PT)	68.29	**76.66**	72.03	64.09	69.20	41.13	**45.40**	62.40
	FLAP	65.54	74.81	70.29	67.48	67.38	38.23	40.00	60.53
	SliceGPT	38.84	64.47	52.34	65.51	59.51	36.86	39.20	50.96
	ASVD	70.34	68.01	53.41	60.93	59.72	32.00	36.60	54.43
	LoRD	75.05	73.88	63.08	**69.46**	69.78	39.16	38.60	61.29
	Bolaco (5 × 1)	79.20	74.97	65.23	67.32	72.35	39.25	41.20	62.79
	Bolaco (5 × 1 w/ PT)	78.78	76.17	73.04	68.51	**74.75**	43.60	44.00	65.55
	Bolaco (5 × 4)	80.24	74.48	66.77	69.14	72.18	41.13	41.00	63.56
	Bolaco (5 × 4 w/ PT)	**80.40**	**76.66**	**73.42**	69.06	73.74	**45.14**	43.40	**65.97**

Table 5: Zero-shot performance of the compressed LLaMA-v2-13b models. w/ PT means the method with post-training. **Bold** denotes the best result at the same compression ratio, while underline indicates the second best result.

Ratio	Methods	BoolQ	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Average
0%	Mistral-7B-v0.1	83.67	80.52	81.03	73.80	80.85	54.01	43.8	71.10
20%	LLM-Pruner	70.06	**77.31**	**72.50**	68.35	69.11	38.23	**41.80**	62.48
	LORD	73.82	74.86	65.53	69.22	71.55	41.13	36.20	61.76
	Bolaco (5 × 1)	74.13	76.01	66.26	69.69	74.24	42.15	39.40	63.13
	Bolaco (5 × 4)	**77.58**	76.12	67.44	**70.09**	**74.96**	**42.41**	39.40	**64.00**

Table 6: Zero-shot performance of the compressed Mistral-7B-v0.1 models. **Bold** denotes the best result at the same compression ratio, while underline indicates the second best result.

mlp_gate, mlp_up, and mlp_down, respectively. 'NA' denotes that the parameter is not compressed.

F Language Modeling Capabilities for Compressed Models

Figure 6 illustrates the perplexity changes on WikiText, PTB, and C4 datasets for different compression methods on LLaMA-v2-7b and 13b as the compression rate increases.

G Case Study

We showcase the generation results of the LLaMA-v2-7b and its compression model via Bolaco in Table 9. We observe that models compressed via Bolaco tend to produce brief and repetitive responses to prompts without post-training. However, this issue can be resolved after efficient post-training, resulting in smooth and informative replies.

G.1 The Generalization of Validation Data

To verify the generalizability of the Bayesian optimization used in Bolaco across various validation data, we sample subsets from the Wikitext, C4, ArXiv, and Wikipedia pre-training datasets to serve as Bolaco's validation data. Table 10 presents the results of Bolaco on these validation data at 20% compression ratio. We observe that models optimized on different validation data exhibit different performance on a single test set, particularly in language modeling capabilities, likely due to the diverse linguistic features of the validation data. However, the average performance across multiple common sense reasoning datasets remains nearly identical, demonstrating the robustness of our method in general capabilities across different validation data.

Method	Ratio	#Params	MACs	Memory
LLaMA 2-7b	0%	6.74B	423.98G	12.62GiB
LLM-Pruner	20%	5.42B	340.48G	10.16GiB
FLAP	20%	5.45B	342.30G	10.22GiB
LoRD	20%	5.45B	370.12G	10.32GiB
Bolaco (5×1)	20%	5.44B	388.95G	10.28GiB
Bolaco (5×4)	20%	5.44B	391.18G	10.25GiB
LLM-Pruner	30%	4.84B	302.83G	9.17GiB
FLAP	30%	4.80B	300.72G	9.04GiB
LoRD	30%	4.79B	341.91G	9.07GiB
Bolaco (5×1)	30%	4.79B	359.48G	9.04GiB
Bolaco (5×4)	30%	4.80B	356.03G	9.06GiB
LLaMA 2-13b	0%	13.02B	824.26G	24.45GiB
LLM-Pruner	20%	10.48B	662.95G	19.75GiB
FLAP	20%	10.48B	663.85G	19.64GiB
LoRD	20%	10.49B	717.86G	19.79GiB
Bolaco (5×1)	20%	10.48B	777.58G	19.71GiB
Bolaco (5×4)	20%	10.48B	772.16G	19.69GiB
LLM-Pruner	30%	9.21B	581.40G	17.35GiB
FLAP	30%	9.21B	582.72G	17.29GiB
LoRD	30%	9.21B	663.15G	17.38GiB
Bolaco (5×1)	30%	9.21B	708.16G	17.36GiB
Bolaco (5×4)	30%	9.21B	694.58G	17.35GiB

Table 7: Statistics of the compressed model.

Model	Method	Ratio	Low rank allocation
LLaMA-v2-7b	Bolaco (5×1)	20%	[744, 1616, 2512, 2408, NA]
	Bolaco (5×4)	20%	[[680, 1728, 2960, NA, NA], [968, 1888, 2536, 2640, 2632], [408, 1488, NA, 2272, 2864], [656, 496, 2824, 2448, 2280]]
	Bolaco (5×1)	30%	[656, 1392, 2128, 2352, 2312]
	Bolaco (5×4)	30%	[[1016, 1632, 2376, 2384, 2384], [840, 1632, 2384, 2376, 2384], [408, 992, 2376, 2384, 2384], [408, 560, 2384, 1896, 1792]]
LLaMA-v2-13b	Bolaco (5×1)	20%	[696, 1920, 2304, NA, 2504]
	Bolaco (5×4)	20%	[[792, 1696, 2864, 2880, 2976], [944, 1440, 2512, 2296, 2920], [656, 1112, 2496, 2480, 2912], [1312, 904, 2264, NA, 1960]]
	Bolaco (5×1)	30%	[512, 1264, 2384, 2328, 2304]
	Bolaco (5×4)	30%	[[528, 1536, 2384, 2376, 2384], [1232, 1624, 2376, 2352, 2344], [800, 1624, 2064, 2368, 2344], [408, 408, 2352, 1936, 1680]]

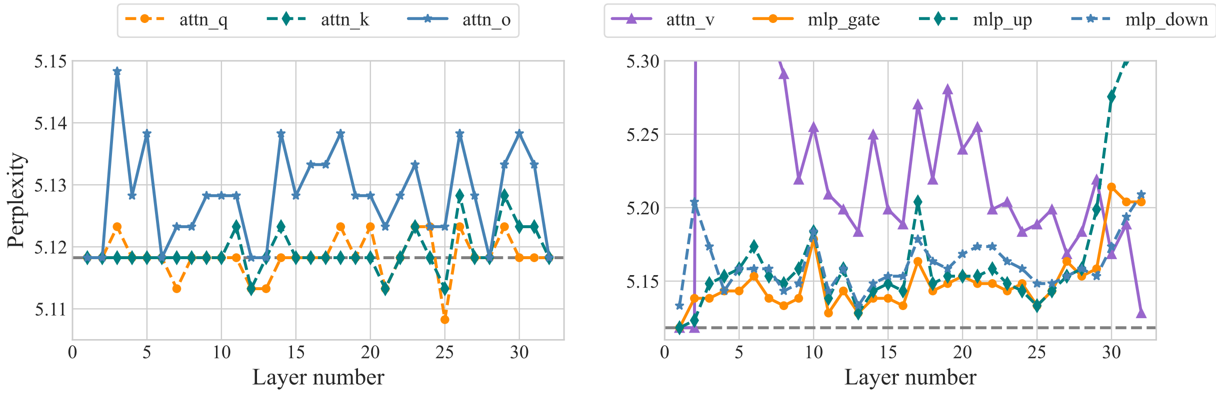
Table 8: The low-rank allocation of our Bolaco.

Model	Generated Sentences
LLaMA 2-7b	Instruction: I sit in front of a computer all day. How do I manage and mitigate eye strain? Response: Eye strain is a common complaint among computer users. It is caused by the eye's inability to focus on the computer screen. This is due to the fact that the computer screen is too close to the eyes. The eyes are also strained by the glare from the computer screen. The best way to manage and mitigate eye strain is to take frequent breaks from the computer. This will allow the eyes to rest and relax.
	Instruction: Translate the sentence 'I have no mouth but I must scream' into Spanish. Response: No tengo boca pero debo gritar.
	Instruction: Who is Donald Trump? Response: Donald Trump is the 45th President of the United States.
Bolaco (20%) (w/o tune)	Instruction: I sit in front of a computer all day. How do I manage and mitigate eye strain? Response: Most of the eye strain in this population is from staring at a computer monitor, which does not affect eye strain nearly as much as looking at a printed page. I recommend that people sit with their computer at arm's length, in a room that has adequate lighting, and not looking at a monitor for prolonged periods. If the screen is too bright, it causes glare and eye strain. If the room is bright, it also causes eye strain. The eyes don't work well with bright light.
	Instruction: Translate the sentence 'I have no mouth but I must scream' into Spanish. Response: Mi boca no has a voz y tengo que scream.
	Instruction: Who is Donald Trump? Response: Trump is the president of the United States.
Bolaco (20%) (w/ tune)	Instruction: I sit in front of a computer all day. How do I manage and mitigate eye strain? Response: Here are some tips to help you manage and mitigate eye strain: 1. Position your monitor correctly: Make sure that your monitor is at least 20-25 inches away from you. If it's too close, your eyes will have to strain to focus on it, which can lead to headaches, eye strain, and fatigue. 2. Use a monitor filter: You can install a monitor filter that blocks ultraviolet rays from your monitor, which can help reduce eye strain and fatigue. 3. Talk to your doctor: If you have been experiencing symptoms of eye strain, headaches, or fatigue, talk to your doctor. He or she may recommend eye drops, glasses, or other solutions to manage your symptoms.
	Instruction: Translate the sentence 'I have no mouth but I must scream' into Spanish. Response: No tengo boca pero debo gritar.
	Instruction: Who is Donald Trump? Response: Donald Trump is a U.S. president who was elected in 2016. He is the 45th president of the United States of America.

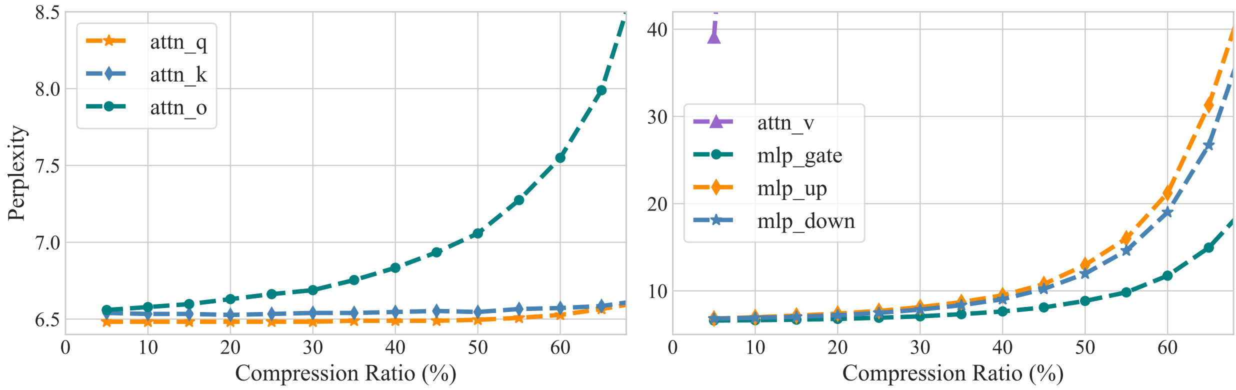
Table 9: Generated Examples from LLaMA-v2-7b and Bolaco.

	Wikitext (↓)	PTB (↓)	Zero-shot (↑)
Wikipedia	7.96	45.84	62.57
Wikitext	7.61	48.37	62.14
C4	7.65	44.56	62.07
Arxiv	8.46	46.77	62.11

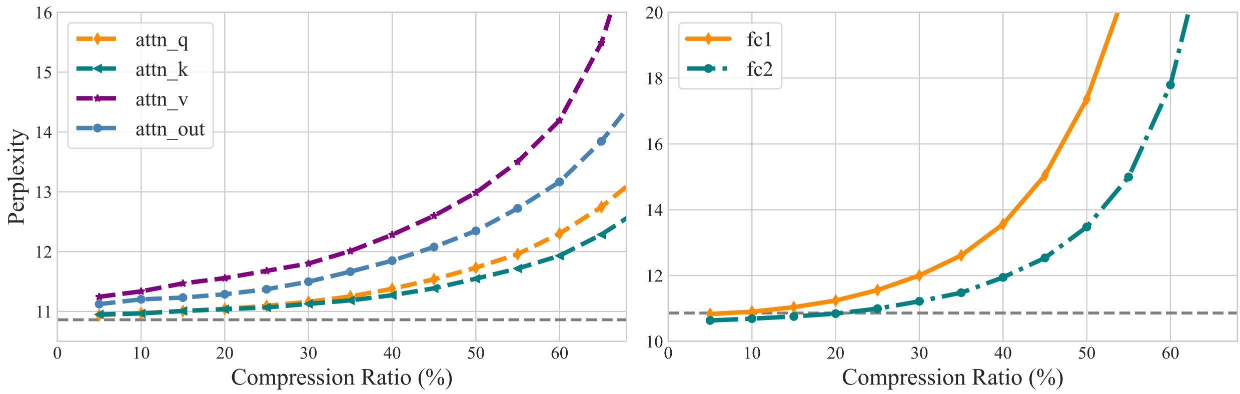
Table 10: Results under different validation data.



(a) Low rank sensitivity of individual layer on LLaMA-v2-7b.

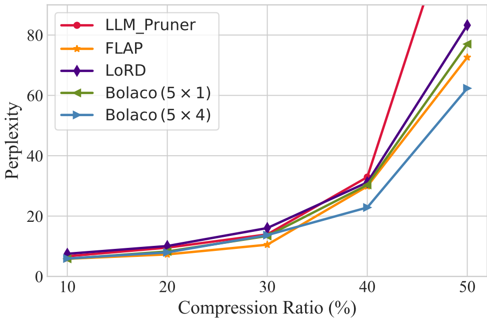


(b) Sensitivity of different types of layers to low-rank compression on the LLaMA-v2-7b-chat.

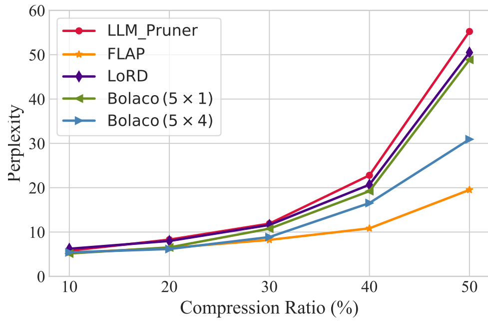


(c) Sensitivity of different types of layers to low-rank compression on the OPT-6.7b.

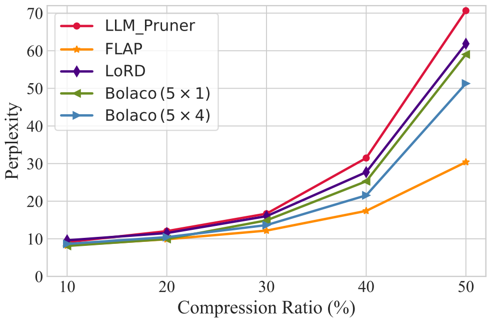
Figure 5: More results on low-rank sensitivity.



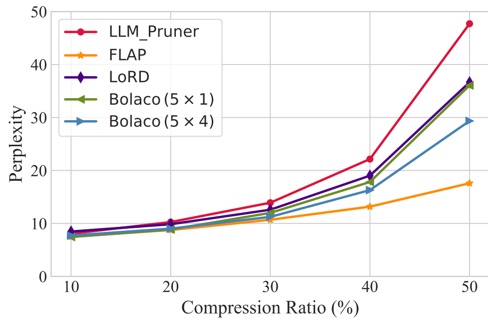
(a) The perplexity of WikiText2 on LLaMA-v2-7b



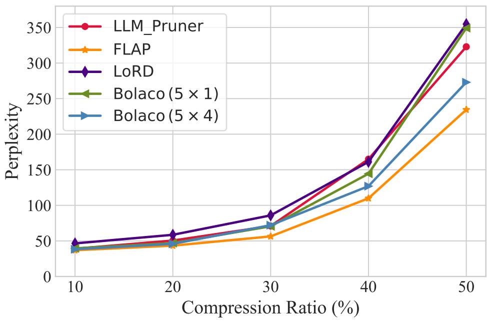
(b) The perplexity of WikiText2 on LLaMA-v2-13b



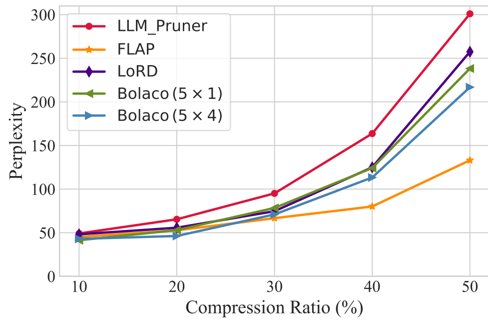
(c) The perplexity of C4 on LLaMA-v2-7b



(d) The perplexity of C4 on LLaMA-v2-13b



(e) The perplexity of PTB on LLaMA-v2-7b



(f) The perplexity of PTB on LLaMA-v2-13b

Figure 6: Language modeling capabilities at different compression ratios.