Integrating Independent Layer-Wise Rank Selection with Low-Rank SVD Training for Model Compression: A Theory-Driven Approach

Yifan Guo[1] , **Alyssa Yu**[2]
[1]Towson University
[2]Poolesville High School
yguo@towson.edu, xiaoyumd@gmail.com

Abstract

In recent years, with the rise of large language models, model sizes have grown dramatically, garnering attention for their remarkable performance but also raising concerns about the substantial computational and communication resources they require. This has created significant challenges in fine-tuning or re-training models on devices with limited computing and memory resources. Efficient model compression through low-rank factorization has emerged as a promising solution, offering a way to balance the tradeoff between compression ratio and prediction accuracy. However, existing approaches to low-rank selection often rely on trial-and-error methods to determine the optimal rank, lacking theoretical guidance and incurring high computational costs. Furthermore, these methods typically treat low-rank factorization as a post-training process, resulting in suboptimal compressed models. In this paper, we design a novel approach by integrating rank selection into the low-rank training process and performing independent layer-wise rank selection under the guidance of a theoretical loss error bound. Specifically, we first conduct a comprehensive theoretical analysis to quantify how low-rank approximations impact the training losses. Building on these insights, we develop an efficient layer-wise rank search algorithm and seamlessly incorporate it into low-rank singular value decomposition (SVD) training. Our evaluation results on benchmark datasets demonstrate that our approach can achieve high prediction accuracy while delivering significant compression performance. Furthermore, our solution is generic and can be extended to broader learning models.

1 Introduction

Deep neural networks have grown increasingly complex and computationally demanding in recent years, often comprising billions or trillions of parameters. Examples include large language models (LLMs) like ChatGPT, computer vision systems like DALL-E, and simulation frameworks like AlphaFold. Without compression, local training or inference on these models typically requires high-performance computing resources and can take days to complete. For instance, training AlexNet on ImageNet originally needed 2-3 days using an NVIDIA K40 GPU in 2012 [Cheng *et al.*, 2018]. More recently, with the surge of LLMs, training a state-of-the-art 7B visual language model can take up to 400 GPU days, let alone even larger models [Liu *et al.*, 2024]. Thus, effectively reducing model size is crucial for scenarios demanding fast re-training (fine-tuning) and inference on resource-constrained edge devices or transferring models between devices with limited computational or communication resources.

The primary challenge in model compression is balancing the tradeoff between reducing model size and maintaining prediction accuracy, ensuring the compressed model does not underfit [Cheng *et al.*, 2018]. Traditionally, three main types of approaches are used: pruning, quantization, and low-rank factorization [Cai *et al.*, 2022]. Particularly, pruning, effective for overparameterized networks, removes unnecessary elements from the weight tensor and is often followed by fine-tuning to minimize the accuracy drop [Han *et al.*, 2015]. Quantization reduces the bit precision of model weights, either through quantization-aware training, where quantization errors are adjusted during training, or post-training quantization, which applies quantization without retraining [Jacob *et al.*, 2018; Cai *et al.*, 2022]. In contrast, low-rank factorization techniques, such as singular value decomposition (SVD), decompose tensors or matrices into simpler components, accelerating inference operations, reducing memory usage in fully connected layers, and adapting well to complex architectures like CNNs, RNNs, and transformers [Yang *et al.*, 2020; Hsu *et al.*, 2022]. Among these techniques, low-rank factorization has attracted more attention recently as it provides better fine-grained control for precise rank adjustments for layer-wise weight matrices, enabling better control over the tradeoff between compression and accuracy.

Despite advances in low-rank factorization, existing approaches [Yang *et al.*, 2020; Kim *et al.*, 2020; Eo *et al.*, 2021; Hsu *et al.*, 2022] still exhibit two major gaps that motivate our work. First, there is no unified principle for selecting the approximated rank that can quantitatively measure how different low-rank constraints degrade training performance. In particular, current methods lack a clear analysis of the core factors influencing the training loss dropping under low-

Approach	Layer-Wise Rank Selection (Independent vs. Dependent)	Search Strategy (Theory-Driven vs. Heuristic)	Rank Selection Timing (During- vs. Post-Training)
[Cheng *et al.*, 2020]	Dependent	Heuristic	Post-Training
[Kim *et al.*, 2020]	Dependent	Theory-Driven	Post-Training
[Eo *et al.*, 2021]	Independent	Heuristic	Post-Training
[Sobolev *et al.*, 2022]	Dependent	Heuristic	Post-Training
[Xiao *et al.*, 2023]	Dependent	Heuristic	Post-Training
[Wang *et al.*, 2023]	Independent	Heuristic	During-Training
[Cao *et al.*, 2024]	Dependent	Heuristic	Post-Training
Ours	Independent	Theory-Driven	During-Training

Table 1: Comparisons with existing rank selection approaches.

rank approximation, leaving practitioners to rely on trial-and-error or exhaustive searches for each layer's weight matrix's rank, posing prohibitively high computational costs. Second, most solutions handle rank selection strictly as a post-training pruning step, overlooking the benefits of jointly optimizing rank selection and network parameters throughout the low-rank training process. These drawbacks highlight the need for a more theoretically grounded approach that systematically integrates rank selection into low-rank training and reduces the search space in rank selections.

In this paper, we design a novel approach for model compression by integrating rank selection into the low-rank training process and performing independent layer. The technical contributions are summarized as follows.

- We provide an in-depth theoretical analysis of how low-rank approximation affects training losses in a quantitative measure.
- We conduct rank selection during training by developing an efficient layer-wise rank search algorithm and incorporating it into low-rank SVD training.
- Our evaluation results on CIFAR-10 and ImageNet datasets verify that our compression approach maintains high prediction accuracies and significant compression performance.

2 Related Work

2.1 Low-Rank Training

Low-rank training often incorporates various penalty terms in the loss function to reduce the rank of weight matrices while preserving high accuracy. For example, [Idelbayev and Carreira-Perpiñán, 2020] introduced a Lagrangian-augmented version of a quadratic penalty term into the loss function. [Idelbayev and Carreira-Perpiñán, 2020] performed two-step low-rank training with alternating updates of the full-rank matrices and their approximated low-rank matrices. [Eo *et al.*, 2021] added a stable rank term to reduce the number of parameters needed during training. Likewise, [Yang *et al.*, 2020] introduced an orthogonality regularization loss and sparsity-inducing regularizers, including the Nuclear and Hoyer norm regularizers. Also, [Huang *et al.*, 2019] introduced altered versions of the Nuclear and Frobenius norms that could potentially be used as penalty terms during training. Furthermore, [Indyk *et al.*, 2019] designed a new training algorithm that produces a learned sketch matrix, reducing the approximation error compared to a random sketch matrix. The low-rank approximation is computed through a projection of the learned sketch matrix.

2.2 Rank Selection

In prior research, rank selection is typically performed on pre-trained deep neural networks to identify the optimal rank based on criteria such as the trade-off between compression and accuracy in the layer-wise matrix and/or tensor decompositions. For example, [Eo *et al.*, 2021] proposed a modified beam search strategy, utilizing both greedy and full-search methods for rank determination. [Sobolev *et al.*, 2022] introduced a novel proxy rank selection metric applicable across multiple types of networks. [Cheng *et al.*, 2020] developed an iterative rank selection process for pre-trained model layers, incorporating an action-reward system to guide decisions. [Kim *et al.*, 2020] used a Bayesian statistical model to determine a globally optimal rank. In addition, [Xiao *et al.*, 2023] designed an algorithm that considers hardware constraints while exploring various rank search spaces. [Wang *et al.*, 2023] approached rank selection differently by focusing on stabilizing layer ranks, identifying the optimal epoch to transition from full-rank to low-rank training. [Cao *et al.*, 2024] proposed a method for jointly learning model weights and decomposition ranks of tensorized neural networks using an ℓ_0-norm based probabilistic approximation.

Rethinking this problem, we categorize existing rank selection approaches along three key dimensions as shown in Table 1: (1) ***Independent vs. Dependent Layer-Wise Rank Selection.*** This measures whether determining the optimal rank for a layer's weight matrix depends on the rank choices in other layers. Independent selections generally require fewer layer-wise iterations in rank selection throughout the whole network, thereby incurring lower computational costs than dependent ones. (2) ***Theory-Driven vs. Heuristic Rank Search Strategy.*** Most existing approaches perform rank selection by relying on heuristic observations on how different rank values affect prediction performance, without an in-depth analysis of how the choice of k in rank-k matrix approximations would theoretically incur the increase in training loss. (3) ***During-Training vs. Post-Training Rank Selection.*** Traditionally, rank selection happens after the training is complete, which would incur increased training loss and reduced prediction precision. Thus, methods to effectively merge rank selection into the training procedure could deserve more investigation to help maximally mitigate these drawbacks. This motivates us to consider integrating rank selection into the low-rank training process and performing

independent layer-wise rank selection under the guidance of theoretical loss error bounds.

3 Preliminaries

3.1 Problem Formulation

We consider a neural network of L layers, denoted by $f_{\boldsymbol{\mathcal{W}}}(\cdot)$, which is parameterized by $\boldsymbol{\mathcal{W}}$, i.e., $\boldsymbol{\mathcal{W}} = \{\boldsymbol{W}^1, \boldsymbol{W}^2, \ldots, \boldsymbol{W}^L\}$, and can be represented as the following cascaded computation form, with the input $X^0 \in \mathbb{R}^{d_0}$: $X^l = a^l(\boldsymbol{W}^l X^{l-1})$ for $l = 1, 2, \ldots, L$. Particularly, $\boldsymbol{W}^l, a^l, X^l$ represent the layer-wise weight matrix, non-linear activation function, and the output of the l-th layer for all l, where $\boldsymbol{W}^l \in \mathbb{R}^{d_l \times d_{l-1}}$, and $X^l \in \mathbb{R}^{d_l}$. Given the training dataset $\mathcal{D}^{tr}$, e.g., $\mathcal{D}^{tr} = \{X_i^0, y_i\}_{i=1}^R$, where $X_i^0 \in \mathbb{R}^{d_0}$ and $y_i \in \mathbb{R}^{d_L}$ denote the i-th input-output pairs in $\mathcal{D}^{tr}$, we define the empirical loss on dataset $\mathcal{D}^{tr}$ as $L(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr}) = \frac{1}{R}\sum_{i=1}^R g(f_{\boldsymbol{\mathcal{W}}}(X_i^0), y_i)$, where loss function g is commonly chosen to be the cross entropy and the squared ℓ_2 norm in classification and regression problems, respectively.

3.2 Low-Rank Training

Strategy 1: Two-Step Low-Rank Training

This strategy aims to search a series of low-rank matrices, e.g., $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}^1, \boldsymbol{\Theta}^2, \ldots, \boldsymbol{\Theta}^L\}$, to approximate each dense matrix in $\boldsymbol{W}$, each with a matrix rank no more than r. Reflecting on the loss function, it would become:

$$\overline{L(\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta}; \mathcal{D}^{tr})} = \frac{1}{R}\sum_{i=1}^R g(f_{\boldsymbol{\mathcal{W}}}(X_i^0), y_i) + \lambda \sum_{l=1}^L ||\boldsymbol{\Theta}^l - \boldsymbol{W}^l||_F$$
$$s.t. \quad \text{rank}(\boldsymbol{\Theta}^l) \leq r, \forall l \in [1, 2, \ldots, L],$$

where λ is the control hyper-parameter and $||\cdot||_F$ denotes the Frobenius norm. To solve this, [Idelbayev and Carreira-Perpiñán, 2021] proposed the learning-compression (LC) algorithm by alternatively updating $\boldsymbol{W}$ or $\boldsymbol{\Theta}$ while fixing the other. It can be decomposed by a "learning" step to update $\boldsymbol{W}$ and ensure learning convergence and a "compression" step to optimize $\boldsymbol{\Theta}$ based on the updated $\boldsymbol{W}$ to achieve the low-rank requirements. However, a key limitation of this two-step approach is the misalignment of optima between the "learning" and "compression" steps, which can hinder overall model efficiency and convergence.

Strategy 2: One-Step Low-Rank Training

Enabling simultaneous optimization for both accuracy and compression, this strategy aims to ensure model convergence and low-rank properties in an integrated way by leveraging different low-rank regularization during training, e.g., $\mathcal{R}(\boldsymbol{W}^l)$, as:

$$\overline{L(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr})} = \underbrace{\frac{1}{R}\sum_{i=1}^R g(f_{\boldsymbol{\mathcal{W}}}(X_i^0), y_i)}_{\overline{L_T(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr})}} + \lambda \underbrace{\sum_{l=1}^L \mathcal{R}(\boldsymbol{W}^l)}_{\overline{L_R(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr})}}.$$

Particularly, the common choices of $\mathcal{R}(\boldsymbol{W}^l)$ include: the ℓ_0-matrix norm $||\boldsymbol{W}^l||_0$, the ℓ_1-matrix norm (Nuclear norm) $||\boldsymbol{W}^l||_1$, and the Hoyer norm $||\boldsymbol{W}^l||_1/||\boldsymbol{W}^l||_F$. Here, since ℓ_0 is non-differentiable and non-smooth, a simple and conventional way is to relax ℓ_0 to ℓ_1. $\overline{L_T(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr})}$ and $\overline{L_R(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr})}$ denote the learning and regularization loss each. In what follows, we define full-rank training as the condition where $\mathcal{R}(\boldsymbol{W}^l) = 0$.

3.3 SVD Matrix Factorization and Low-Rank Approximations

Applying SVD, a matrix $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ can be decomposed as $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times r}$ ($\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal), $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$ is a diagonal matrix, i.e., $\boldsymbol{\Sigma} = \text{diag}(\sigma)$, $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_r]$, and $\sigma_k (k \in [r])$ are singular values of $\boldsymbol{W}$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Here, r is the rank of $\boldsymbol{W}$ and $r \leq \min\{m, n\}$. Similarly, we can decompose $\boldsymbol{W}$ as the product of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, where $\boldsymbol{W}_1 = \boldsymbol{U}\sqrt{\boldsymbol{\Sigma}}$ and $\boldsymbol{W}_2 = \sqrt{\boldsymbol{\Sigma}}\boldsymbol{V}^T$. By choosing an appropriate k, we have the corresponding low-rank matrix $\boldsymbol{W}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T$, where $\boldsymbol{U}_k, \boldsymbol{\Sigma}_k, \boldsymbol{V}_k$ are top-k vectors truncated from $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$.

4 Our Approach

4.1 A Theoretical Analysis of Loss Error Bounds Under Layer-Wise Rank-k Approximation

We first provide an in-depth theoretical analysis of how low-rank approximation affects predictive accuracy in a quantitative measure. By examining the key factors that influence loss error bounds in traditional classification and regression problems resulting from low-rank approximations, we derive practical principles for selecting an appropriate rank. To this end, we investigate the discrepancy in the output layer between the full-rank and low-rank parameter spaces when given the same inputs, as illustrated in Fig. 1 and quantified in Theorem 1.

Theorem 1 (The output difference bound for rank-k approximation over L-layer neural networks)**.** *We denote a^l to be the activation function for the l-th layer, and assume a^l is ρ_l-Lipschitz and $a^l(0) = 0$ for all $l \in [1, L]$. Let X^0 be the initial input vector, X^l and X_k^l be the output vectors as a result of passing the full-rank matrix $\boldsymbol{W}^l$ and low-rank matrix $\boldsymbol{W}_k^l$ through the l-th layer, respectively, and σ_i^l be the i-th singular value of $\boldsymbol{W}^l$. We define k^l such that the top k^l largest singular values of the full-rank matrix $\boldsymbol{W}^l$ are kept in the corresponding low-rank SVD approximated matrix $\boldsymbol{W}_k^l$ in layer l. Then, the output difference from rank-k approximation over L-layer feed-forward networks $||X^L - X_k^L||_2$ is upper-bounded by $\left(\prod_{l=1}^L \rho_l \sigma_1^l\right)\left(\sum_{l=1}^L \frac{\sigma_{k^l+1}^l}{\sigma_1^l}\right)||X^0||_2$.*

We then present two extended results on loss error bounds for rank-k approximations in classification and regression tasks, as stated in Theorems 2 and 3, respectively.

Theorem 2 (The loss error bound from rank-k approximation in classification problems)**.** *Following the settings in Theorem 1, we consider a C-class classification problem. Let $X_i^L \in \mathbb{R}^C$ and $X_{i,k}^L \in \mathbb{R}^C$ be the output logits when feeding a input X_i^0 sampled from the training dataset $\mathcal{D}^{tr}$, e.g., $\mathcal{D}^{tr} = \{X_i^0, y_i\}_{i=1}^R$, from the full-rank parameter space $\boldsymbol{\mathcal{W}}$ and*

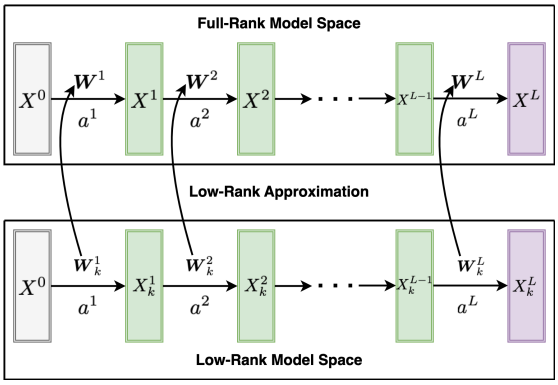


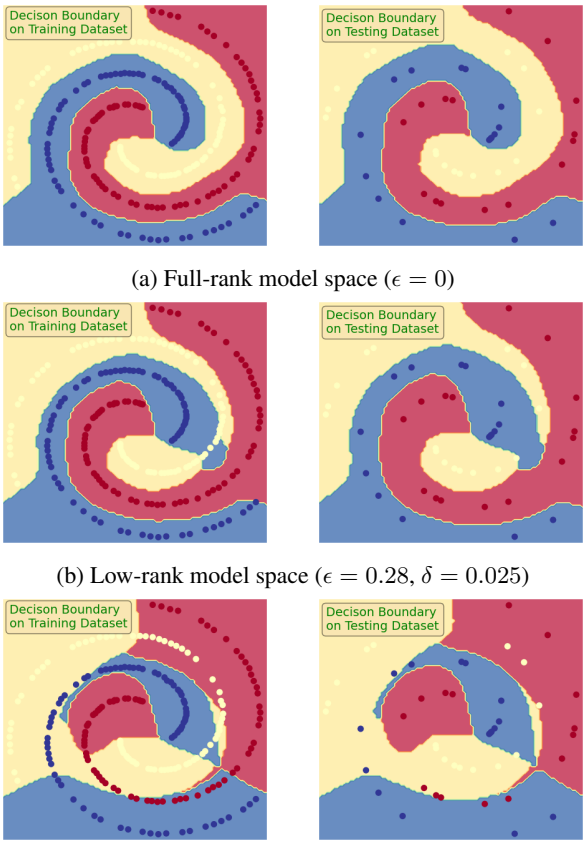
Figure 1: An illustration of our focus: the output difference bound caused by rank-k approximation over L-layer feed-forward networks.

low-rank parameter space $\boldsymbol{\mathcal{W}_k}$*, respectively. Particularly,* $\boldsymbol{\mathcal{W}} = \{\boldsymbol{W}^1, \boldsymbol{W}^2, \ldots, \boldsymbol{W}^L\}$, $\boldsymbol{\mathcal{W}_k} = \{\boldsymbol{W}_k^1, \boldsymbol{W}_k^2, \ldots, \boldsymbol{W}_k^L\}$, $X_i^L = f_{\boldsymbol{\mathcal{W}}}(X_i^0)$, $X_{i,k}^L = f_{\boldsymbol{\mathcal{W}_k}}(X_{i,k}^0)$, *and* $||X_i^0||_2 \leq B$, *for* $\forall\ i \in [1, R]$. *Let* $z_i = \texttt{softmax}(X_i^L)$ *and* $z_{i,k} = \texttt{softmax}(X_{i,k}^L)$, *where* `softmax` *is the softmax function. We consider the cross-entropy function as the loss function, i.e.,* $g(z, y) = -y^T \log(z)$. *Let* $L(\boldsymbol{\mathcal{W}}; X_i^0) = g(z_i, y_i)$ *and* $L(\boldsymbol{\mathcal{W}_k}; X_i^0) = g(z_{i,k}, y_i)$. *Now, we set* $\frac{\sigma_{k^l+1}^l}{\sigma_1^l} < \delta, \forall\ l \in [1, L]$. *Then, we have, for* $\forall\ \epsilon > 0, \exists\ \delta = \frac{\epsilon}{\sqrt{2}BL(\prod_{l=1}^L \rho_l \sigma_1^l)}$, *s.t.* $|L(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr}) - L(\boldsymbol{\mathcal{W}_k}; \mathcal{D}^{tr})| < \epsilon$.

Theorem 3 (The loss error bound from rank-k approximation in regression problems)**.** *Following the settings in Theorem 1, we consider a regression problem. Let* X_i^L *and* $X_{i,k}^L$ *be the outputs at layer* L *when feeding a input* X_i^0 *sampled from the training dataset* $\mathcal{D}^{tr}$*, e.g.,* $\mathcal{D}^{tr} = \{X_i^0, y_i\}_{i=1}^R$, *from the full-rank parameter space* $\boldsymbol{\mathcal{W}}$ *and low-rank parameter space* $\boldsymbol{\mathcal{W}_k}$*, respectively. Particularly,* $\boldsymbol{\mathcal{W}} = \{\boldsymbol{W}^1, \boldsymbol{W}^2, \ldots, \boldsymbol{W}^L\}$, $\boldsymbol{\mathcal{W}_k} = \{\boldsymbol{W}_k^1, \boldsymbol{W}_k^2, \ldots, \boldsymbol{W}_k^L\}$, $X_i^L = f_{\boldsymbol{\mathcal{W}}}(X_i^0)$, $X_{i,k}^L = f_{\boldsymbol{\mathcal{W}_k}}(X_{i,k}^0)$, *and* $||X_i^0||_2 \leq B$, *for* $\forall\ i \in [1, R]$. *We consider the loss function as* $g(z, y) = ||z - y||_2$. *Let* $L(\boldsymbol{\mathcal{W}}; X_i^0) = g(X_i^L, y_i)$ *and* $L(\boldsymbol{\mathcal{W}_k}; X_i^0) = g(X_{i,k}^L, y_i)$. *Now, we set* $\frac{\sigma_{k^l+1}^l}{\sigma_1^l} < \delta, \forall\ l \in [1, L]$. *Then, we have, for* $\forall\ \epsilon > 0, \exists\ \delta = \frac{\epsilon}{BL(\prod_{l=1}^L \rho_l \sigma_1^l)}$, *s.t.* $|L(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr}) - L(\boldsymbol{\mathcal{W}_k}; \mathcal{D}^{tr})| < \epsilon$.

The results in Theorem 2 and 3 indicate the principle to adjust the optimal k to maintain the loss error bound. For instance, if the tolerance of the maximum loss decrease is ϵ, then we can always find the corresponding optimal rank k^l for the l-th layer matrix, ensuring that k^l is the largest index whose singular value satisfies $\frac{\sigma_{k^l}^l}{\sigma_1^l} \geq \delta, \forall\ l \in [L]$. Here, δ is fixed when ϵ, ρ_l, L, B, and σ_1^l are fixed values. Furthermore, it is noteworthy that the determination of the optimal k^l in our method is layer-wise independent, meaning it does not rely on the optimal rank information of other layers.

To validate the feasibility of identifying δ based on our derived ϵ-δ correlation and determining the optimal k^l, we conduct a pilot study using a simple 3-layer feed-forward neural network for a ternary classification problem. Fig. 2 visualizes the decision boundaries for each model space. Particularly, in the first row, the decision boundary under the initial full-rank model space clearly separates the data classes correctly. However, as the loss error bound ϵ increases, the corresponding δ also increases, indicating that more information is truncated as smaller values of k^l are assigned to each layer. This truncation leads to underfitting in the low-rank model, preventing it from effectively capturing the patterns of the original full-rank model. Thus, we aim to identify the key turning point where the smallest k^l is sufficient to retain the full-rank model's representational capacity.



(a) Full-rank model space ($\epsilon = 0$)

(b) Low-rank model space ($\epsilon = 0.28$, $\delta = 0.025$)

(c) Low-rank model space ($\epsilon = 0.33$, $\delta = 0.03$)

Figure 2: A visualization of the decision boundaries on the training dataset (left column) and testing dataset (right column).

4.2 Independent Layer-Wise Rank Selection

Due to the complex non-linear structure in complex CNN models, such as ResNets or DenseNets, calculating the exact Lipschitz constant of layer-wise parameters is computationally expensive and often requires time-consuming approximation techniques like Jacobian analysis [Bhowmick *et al.*, 2021] or layer-wise local estimations [Herrera *et al.*, 2020]. These challenges hinder the extension of theoretical findings from simpler feed-forward networks to more complex CNN architectures.

To address this and facilitate efficient determination of an

Algorithm 1: Our Rank Selection Algorithm

Input: full-rank parameters $\boldsymbol{\mathcal{W}}$, loss error tolerance ϵ, stop searching precision $\Delta\delta$
Output: low-rank parameters $\boldsymbol{\mathcal{W}_k}$

Function `RankSelection`($\boldsymbol{\mathcal{W}}$, ϵ, $\Delta\delta$)**:**
  $floss \leftarrow \overline{L_T(\boldsymbol{\mathcal{W}}; \mathcal{D}^{tr})}$;
  $l \leftarrow 0; u \leftarrow 1; \delta \leftarrow (l+u)/2$;
  while $|l-u| \geq \Delta\delta$ *or* $|floss - loss| \geq \epsilon$ **do**
    $\boldsymbol{\mathcal{W}_k} \leftarrow$ `SVDLowRankApprox`$(\boldsymbol{\mathcal{W}}, \delta)$;
    $loss \leftarrow \overline{L_T(\boldsymbol{\mathcal{W}_k}; \mathcal{D}^{tr})}$;
    if $|floss - loss| < \epsilon$ **then**
      $l \leftarrow \delta; \delta \leftarrow (l+u)/2$;
    else
      $u \leftarrow \delta; \delta \leftarrow (l+u)/2$;
  return $\boldsymbol{\mathcal{W}_k}$;

Algorithm 2: SVD Low-Rank Approximation

Function `SVDLowRankApprox`($\boldsymbol{\mathcal{W}}$, δ)**:**
  $\boldsymbol{\mathcal{W}_k} \leftarrow \emptyset$;
  for *each* $\boldsymbol{W}^l \in \boldsymbol{\mathcal{W}}$ **do**
    $\boldsymbol{U}^l, \boldsymbol{\Sigma}^l, \boldsymbol{V}^l \leftarrow$ `SVD`$(\boldsymbol{W}^l)$;
    $k^l \leftarrow \text{argmax}_k\{k|\sigma_k^l/\sigma_1^l \geq \delta\}$;
    $\boldsymbol{W}_k^l \leftarrow \boldsymbol{U}_k^l \boldsymbol{\Sigma}_k^l \boldsymbol{V}_k^l; \boldsymbol{\mathcal{W}_k} \leftarrow \boldsymbol{\mathcal{W}_k} \cup \boldsymbol{W}_k^l$;
    // $\boldsymbol{U}_k^l, \boldsymbol{\Sigma}_k^l, \boldsymbol{V}_k^l$ are top-k^l vectors truncated from $\boldsymbol{U}^l, \boldsymbol{\Sigma}^l, \boldsymbol{V}^l$
  return $\boldsymbol{\mathcal{W}_k}$

appropriate rank k^l, we develop a binary search-based rank selection algorithm inspired by the results of Theorems 2 and 3, detailed in Algorithms 1 and 2. It is designed to identify an optimal value of δ (on a scale from 0 to 1) such that the difference between the low-rank loss and full-rank loss $floss$ is below ϵ. We repeatedly update the lower and upper bounds of δ, i.e., l and u, and repeatedly halve the search space until the searching precision $\Delta\delta$ has been reached. Assuming the time complexity of function `SVDLowRankApprox` is T, the time complexity of Algorithm 1 is $\mathcal{O}(T\log(1/\Delta\delta))$. In practice, we observe that when $\Delta\delta$ is sufficiently small and the condition $|floss - loss| < \epsilon$ is satisfied, we stop the search early to avoid unnecessary computations.

4.3 Our Integrated Solution: Rank Selection Enabled Low-Rank SVD Training

Traditionally, rank selection only happens after the training is complete, which does not fully utilize the benefits of during-training rank selections, incurring increased training loss and reduced prediction precision. Thus, we merge rank selection into the training by integrating Algorithm 1 into low-rank SVD training.

Being aware of the potentially high computational cost associated with performing SVD after model training, to address this, we adopt low-rank SVD training [Yang *et al.*, 2020]. It directly optimizes $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{\Sigma}$ as the trainable parameters, replacing the original kernel $\mathbf{K}$ or weight matrix $\mathbf{W}$ in the network, thereby avoiding repeated SVD computations after training, in each layer.

Particularly, during the forward pass, $\mathbf{K}$ or $\mathbf{W}$ are converted into the form of two consecutive layers as follows. For example, for a convolution layer, the kernel $\mathbf{K} \in \mathbb{R}^{n\times c\times w\times h}$ can be represented as a 4-D tensor, where n, c, w, h represent the number of filters, the number of input channels, and the width and the height of the filter, respectively. Then, we can employ either channel-wise decomposition or spatial-wise decomposition to decompose the convolution layer. Specifically, channel-wise decomposition first reshapes $\mathbf{K}$ to a 2-D matrix $\hat{\mathbf{K}} \in \mathbb{R}^{n\times cwh}$ and decomposes it using SVD into $\mathbf{U} \in \mathbb{R}^{n\times r}$, $\mathbf{V} \in \mathbb{R}^{cwh\times r}$ and $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $r = \min(n, cwh)$. Thus, $\mathbf{K}$ is decomposed into two consecutive layers with the kernels of $\mathbf{K_1} \in \mathbb{R}^{r\times c\times w\times h}$ and $\mathbf{K_2} \in \mathbb{R}^{n\times r\times 1\times 1}$. Similarly, spatial-wise decomposition reshapes $\mathbf{K}$ to a 2-D matrix $\hat{\mathbf{K}} \in \mathbb{R}^{nw\times ch}$ and decomposes it into $\mathbf{U} \in \mathbb{R}^{nw\times r}$, $\mathbf{V} \in \mathbb{R}^{ch\times r}$ and $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$ with $r = \min(nw, ch)$. Thus, the decomposed layers would have kernels $\mathbf{K_1} \in \mathbb{R}^{r\times c\times 1\times h}$ and $\mathbf{K_2} \in \mathbb{R}^{n\times r\times w\times 1}$.

Afterwards, backpropagation and optimization are then performed directly on $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{\Sigma}$ for each layer. This allows for direct access to the singular values $\mathbf{\Sigma}$ without performing the time-consuming SVD as in Algorithm 2.

Thus, our training parameter space would become: $\boldsymbol{\mathcal{U}} = \{\mathbf{U}^1, \mathbf{U}^2, \ldots, \mathbf{U}^L\}$, $\boldsymbol{\Sigma} = \{\mathbf{\Sigma}^1, \mathbf{\Sigma}^2, \ldots, \mathbf{\Sigma}^L\}$, and $\boldsymbol{\mathcal{V}} = \{\mathbf{V}^1, \mathbf{V}^2, \ldots, \mathbf{V}^L\}$. The corresponding loss function would become:

$$L(\boldsymbol{\mathcal{U}}, \boldsymbol{\Sigma}, \boldsymbol{\mathcal{V}}; D^{tr}) = L_T(\boldsymbol{\mathcal{U}}, \boldsymbol{\Sigma}, \boldsymbol{\mathcal{V}}) + \lambda_O L_O(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}}) + \lambda_R L_R(\boldsymbol{\Sigma})$$

Here, L_T represents the training loss, which satisfies

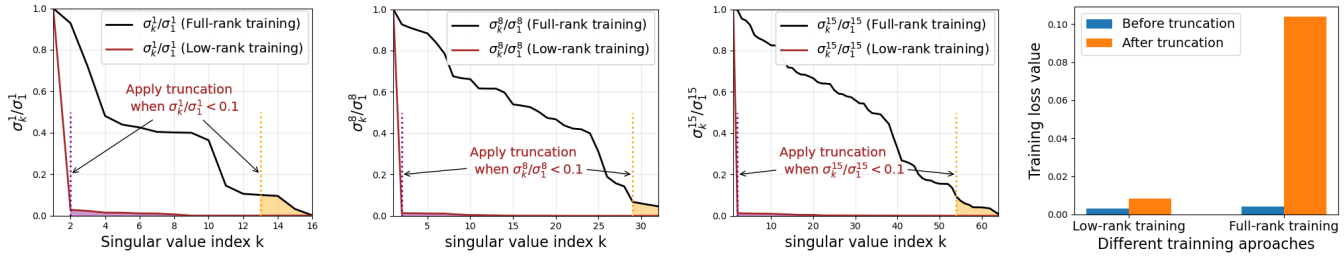
$$L_T(\boldsymbol{\mathcal{U}}, \boldsymbol{\Sigma}, \boldsymbol{\mathcal{V}}) = \frac{1}{R}\sum_{i=1}^{R} g(f_{\boldsymbol{\mathcal{W}}}(X_i^0), y_i),$$

L_O represents the orthogonalization loss, which satisfies

$$L_O(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}}) = \sum_{l=1}^{L} \frac{||(\mathbf{U}^l)^T\mathbf{U}^l - \mathbf{I}||_F + ||(\mathbf{V}^l)^T\mathbf{V}^l - \mathbf{I}||_F}{(r^l)^2},$$

and L_R represents the low-rank regularization loss. Particularly, for each $\mathbf{W}^l \in \boldsymbol{\mathcal{W}}$, we have $\mathbf{W}^l = \mathbf{U}^l\mathbf{\Sigma}^l(\mathbf{V}^l)^T$. For the low-rank regularization term, we consider the following two penalty functions: (1) Nuclear norm $L_R(\mathbf{\Sigma}) = \sum_{l=1}^{L}\left(\sum_{i=1}^{r^l}\sigma_i^l\right)$; (2) Squared Hoyer norm $L_R(\mathbf{\Sigma}) = \sum_{l=1}^{L}\frac{\left(\sum_{i=1}^{r^l}\sigma_i^l\right)^2}{\sum_{i=1}^{r^l}(\sigma_i^l)^2}$. $\boldsymbol{I}$ is the identity matrix. λ_O and λ_R are hyperparameters that control the trade-off between the orthogonality loss and the sparsity-inducing regularization loss, respectively, and are critical to the training process.

In our design, we consider conducting rank selection after each round of training based on loss function $L(\boldsymbol{\mathcal{U}}, \boldsymbol{\Sigma}, \boldsymbol{\mathcal{V}}; D^{tr})$. During the training process, we repeatedly perform truncations by replacing $\mathbf{U}^l$, $\mathbf{\Sigma}^l$, $\mathbf{V}^l$ with the truncated form $\mathbf{U}_k^l$, $\mathbf{\Sigma}_k^l$, $\mathbf{V}_k^l$, which replaces $\mathbf{W}^l$ with $\mathbf{W}_k^l = \mathbf{U}_k^l\mathbf{\Sigma}_k^l(\mathbf{V}_k^l)^T$ after each round of training.



(a) σ_k^l/σ_1^l in the 1^{st} layer (b) σ_k^l/σ_1^l in the 8^{th} layer (c) σ_k^l/σ_1^l in the 15^{th} layer (d) Training loss comparisons

Figure 3: The comparisons between full-rank and low-rank training on the ResNet-20 model.

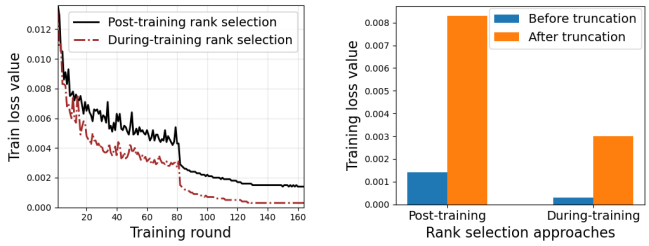
Then, we present two claims to highlight the advantages of low-rank over full-rank training in compression efficiency and the benefits of in-training rank selection, supported by our empirical observations.

Claim 1. *Compared to regular full-rank training, low-rank training achieves better compression performance under the same δ and experiences smaller increases in training loss when post-training matrix truncations are applied.*

As illustrated in Fig. 3, we evaluate the training performance of both low-rank and full-rank training on the benchmark ResNet-20 model using the CIFAR-10 dataset to validate Claim 1. Particularly, we consider channel-wise decomposition and the squared Hoyer norm as the low-rank penalty term. We first depict the curves of σ_k^l/σ_1^l in the 1^{st}, 8^{th}, 15^{th} layers in the model, respectively, in Fig. 3a-3c. We find that when we choose the truncation threshold δ to be 0.1, the matrix (reshaped kernel) rank has a slight reduction in each layer, i.e., from 16 to 13, from 32 to 29, and from 64 to 54, after truncations under full-rank training. In contrast, low-rank training (with low-rank regularization enforcement) leads to a significant rank reduction in each layer, from 16 to 2, 32 to 2, and 64 to 2. This demonstrates the substantially high compression rate achieved with low-rank training when applying rank selection algorithms. Moreover, as shown in Fig. 3d, low-rank training results in smaller increases in training loss, as fewer informative features are truncated (evidenced by smaller cumulative singular values being removed) compared to full-rank training. These findings underscore the importance of low-rank training when applying rank selection.

Claim 2. *Compared with post-training rank selection, during-training rank selection comes with lower training loss errors after post-training matrix truncations.*

Following the same experimental setting in Claim 1, we evaluate the training performance of both post-training and during-training rank selection after post-training truncations to validate Claim 2 as shown in Fig. 4. In particular, we consider performing rank selection immediately after each training round, which ensures that model parameters in the next round are updated based on the truncated model, further reinforcing the model's low-rank structure and sparsity. As demonstrated in Fig. 4b, incorporating rank selection during the training significantly reduces fluctuations in training loss after post-training truncations, highlighting its effectiveness compared to rank selection alone after the training.



(a) Training loss curves (b) Training loss comparisons

Figure 4: The comparisons between post-training and during-training rank selection on the ResNet-20 model.

5 Performance Evaluation

5.1 Experimental Setup

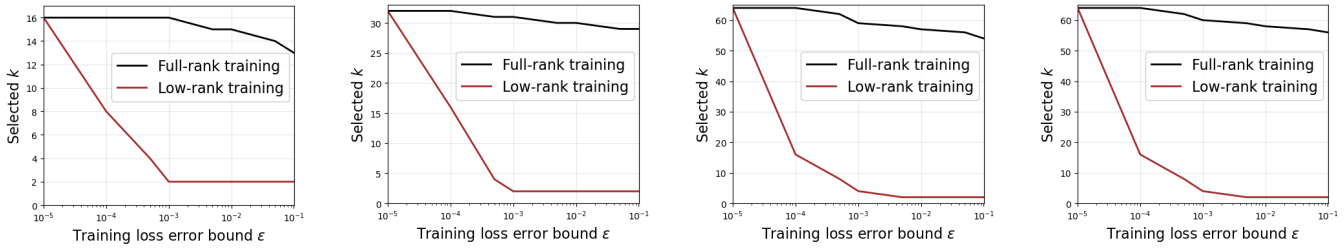
Datasets and Training Models. We conducted our training procedure using models from the ResNet series [He *et al.*, 2016], including ResNet-20, ResNet-32, and ResNet-56 on the CIFAR-10 dataset [Krizhevsky *et al.*, 2009], and ResNet-18 and ResNet-50 on the ImageNet dataset [Russakovsky *et al.*, 2015]. All models were trained on a workstation with two NVIDIA RTX 3090 GPUs.

Hyper-parameters Settings. We set the learning rate to 0.001 with a scheduled learning delay in later training rounds. The batch sizes for training and testing are set to 100 and 1000, respectively. Additionally, λ_O and λ_R are dynamically adjusted based on datasets and low-rank regularizers.

Performance Evaluation Metrics. We consider two key aspects in performance evaluations for each model compression approach, e.g., testing accuracy and compression ratio. Particularly, we define compression ratio CR as in [Eo *et al.*, 2021] as $CR = \frac{\sum_{l=1}^{L} d_l d_{l-1} \mathbb{1}^l + r^l(d_l + d_{l-1})(1-\mathbb{1}^l)}{\sum_{l=1}^{L} d_l d_{l-1}}$, where $\mathbb{1}^l$ is defined as 1 if $d_l d_{l-1} \leq r^l(d_l + d_{l-1})$ and 0 otherwise, and d_l is the output vector's shape at the l-th layer. The smaller CR, the better model compression is achieved. Also, we assess the training and testing loss errors under various rank-k approximations to demonstrate the effectiveness of our approach.

5.2 Correlation Between the Training Loss Error Bound and Rank Selection

We begin with analyzing how the choice of ϵ influences the selected rank k at each layer, as illustrated in Fig. 5, with some selected layers as examples. Our findings reveal that a



(a) The 1^{st} layer (b) The 8^{th} layer (c) The 15^{th} layer (d) The 19^{th} layer

Figure 5: The correlation between the training loss error and rank selection on the ResNet-20 model.

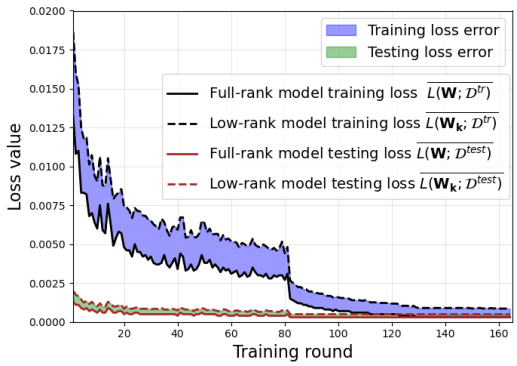


Figure 6: Analysis of during-training rank selections on the ResNet-20 model.

larger training loss error tolerance ϵ allows for a smaller final rank. More importantly, for the same threshold value of ϵ, the truncated matrix rank obtained through low-rank training is substantially smaller than that achieved with standard full-rank training. This highlights the necessity of low-rank training to facilitate rank selections.

5.3 Analysis of During-Training Rank Selections

Next, we conduct an in-depth analysis of the fluctuations introduced by applying rank selection immediately after each round of training. Fig. 6 provides a detailed illustration of these fluctuations under our low-rank SVD training framework, which incorporates rank selection during training. Notably, the training loss error caused by low-rank approximations is higher during the early training stages compared to the later stages. This can be attributed to two key factors: (i) the model becomes progressively sparser as training advances; (ii) the learning rate decay weakens training loss errors over time. Then, the testing loss errors exhibit lower magnitudes than their corresponding training loss errors.

5.4 Effectiveness and Superiority of Our Solution

Finally, we evaluate the performance of various approaches in terms of test accuracy and compression ratio on both the CIFAR-10 and ImageNet datasets, as shown in Table 2 and Table 3, respectively. We find our approach consistently achieves the best performance on both datasets, demonstrating the highest test accuracy and the lowest CR. Additionally, we observe a notable trend: as the model complexity increases, test accuracy improves slightly, accompanied by further reductions in CR. Furthermore, compared to the nuclear norm low-rank regularization term, we find that the squared Hoyer norm shows better performance in both utility and compression efficiency under the same settings.

Approach	ResNet-20		ResNet-32		ResNet-56	
	Test Acc ↑	CR ↓	Test Acc ↑	CR ↓	Test Acc ↑	CR ↓
[Yang *et al.*, 2020] (Channel, Squared Hoyer)	0.887	0.202	0.893	0.367	0.924	0.485
[Yang *et al.*, 2020] (Spatial, Squared Hoyer)	0.866	0.242	0.889	0.413	0.918	0.522
[Wang *et al.*, 2023]	0.822	0.337	0.834	0.398	0.844	0.441
Ours (Channel, Nuclear)	0.870	**0.155**	0.879	**0.312**	0.892	**0.422**
Ours (Spatial, Nuclear)	0.867	0.221	0.873	0.319	0.899	0.438
Ours (Channel, Squared Hoyer)	**0.892**	**0.155**	**0.899**	**0.312**	**0.929**	**0.422**
Ours (Spatial, Squared Hoyer)	0.873	0.221	0.881	0.319	0.912	0.438

Table 2: Performance evaluation on the CIFAR-10 dataset.

Approach	ResNet-18		ResNet-50	
	Test Acc ↑	CR ↓	Test Acc ↑	CR ↓
[Yang *et al.*, 2020] (Channel, Squared Hoyer)	0.684	0.204	0.691	0.392
[Yang *et al.*, 2020] (Spatial, Squared Hoyer)	0.670	0.221	0.678	0.411
Ours (Channel, Squared Hoyer)	**0.690**	**0.181**	**0.696**	**0.366**
Ours (Spatial, Squared Hoyer)	0.672	0.207	0.681	0.398

Table 3: Performance evaluation on the ImageNet dataset.

6 Final Remarks

In this paper, we have designed a novel approach by integrating rank selection into the low-rank training process and performing independent layer-wise rank selection under the theoretical loss error bound guidance. The technical contributions are summarized as follows. We have provided an in-depth theoretical analysis that quantitatively measures how low-rank approximation affects training losses. This analysis is a key aspect of our work. We have also considered rank selection during training by developing an efficient layer-wise rank search algorithm and incorporating it into low-rank singular SVD training. Our evaluation results on CIFAR-10 and ImageNet datasets have verified that our approach maintains high precision and significant compression performance.

Ethical Statement

There are no ethical issues.

References

[Bhowmick *et al.*, 2021] Aritra Bhowmick, Meenakshi D'Souza, and G Srinivasa Raghavan. LipBaB: computing exact lipschitz constant of ReLU networks. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part IV 30*, pages 151–162. Springer, 2021.

[Cai *et al.*, 2022] Han Cai, Ji Lin, and Song Han. Efficient methods for deep learning. In *Advanced Methods and Deep Learning in Computer Vision*, pages 159–190. Elsevier, 2022.

[Cao *et al.*, 2024] Tianxiao Cao, Lu Sun, Canh Hao Nguyen, and Hiroshi Mamitsuka. Learning low-rank tensor cores with probabilistic ℓ_0-regularized rank selection for model compression. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 3780–3788. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Main Track.

[Cheng *et al.*, 2018] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

[Cheng *et al.*, 2020] Zhiyu Cheng, Baopu Li, Yanwen Fan, and Yingze Bao. A novel rank selection scheme in tensor ring decomposition based on reinforcement learning for deep neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3292–3296. IEEE, 2020.

[Eo *et al.*, 2021] Moonjung Eo, Suhyun Kang, and Wonjong Rhee. A highly effective low-rank compression of deep neural networks with modified beam-search and modified stable rank. *CoRR*, abs/2111.15179, 2021.

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

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

[Herrera *et al.*, 2020] Calypso Herrera, Florian Krach, and Josef Teichmann. Local lipschitz bounds of deep neural networks. *arXiv preprint arXiv:2004.13135*, 2020.

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

[Huang *et al.*, 2019] Yan Huang, Guisheng Liao, Yijian Xiang, Lei Zhang, Jie Li, and Arye Nehorai. Low-rank approximation via generalized reweighted iterative nuclear and frobenius norms. *IEEE Transactions on Image Processing*, 29:2244–2257, 2019.

[Idelbayev and Carreira-Perpiñán, 2020] Yerlan Idelbayev and Miguel Á Carreira-Perpiñán. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8049–8059, 2020.

[Idelbayev and Carreira-Perpiñán, 2021] Yerlan Idelbayev and Miguel Á Carreira-Perpiñán. LC: a flexible, extensible open-source toolkit for model compression. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4504–4514, 2021.

[Indyk *et al.*, 2019] Piotr Indyk, Ali Vakilian, and Yang Yuan. *Learning-based low-rank approximations*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[Jacob *et al.*, 2018] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[Kim *et al.*, 2020] Taehyeon Kim, Jieun Lee, and Yoonsik Choe. Bayesian optimization-based global optimal rank selection for compression of convolutional neural networks. *IEEE Access*, 8:17605–17618, 2020.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Liu *et al.*, 2024] Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, et al. NVILA: Efficient frontier visual language models. *arXiv preprint arXiv:2412.04468*, 2024.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[Sobolev *et al.*, 2022] Konstantin Sobolev, Dmitry Ermilov, Anh-Huy Phan, and Andrzej Cichocki. Pars: Proxy-based automatic rank selection for neural network compression via low-rank weight approximation. *Mathematics*, 10(20):3801, 2022.

[Wang *et al.*, 2023] Hongyi Wang, Saurabh Agarwal, Yoshiki Tanaka, Eric Xing, Dimitris Papailiopoulos, et al. Cuttlefish: Low-rank model training without all the tuning. *Proceedings of Machine Learning and Systems*, 5:578–605, 2023.

[Xiao *et al.*, 2023] Jinqi Xiao, Chengming Zhang, Yu Gong, Miao Yin, Yang Sui, Lizhi Xiang, Dingwen Tao, and Bo Yuan. HALOC: hardware-aware automatic low-rank

compression for compact neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10464–10472, 2023.

[Yang *et al.*, 2020] Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Helen Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2899–2908, 2020.