Algorithm 1 Dynamic-rank Training

Require: Model parameters Θ, Training dataset $\mathcal{D}$, Total epochs E, low-rank dimension k, Inflation epoch I, Deflation epoch D, Learning rate schedule η.

1: Initialize model parameters Θ in a low-rank form.
2: **for** $t = 1 \rightarrow E$ **do**
3: **if** $t = I$ **then**
4: *// Inflate model to full-rank*
5: **for** low-rank weight $(\mathbf{A}, \mathbf{B})$ with base $\mathbf{W}_0$ in Θ **do**
6: $\mathbf{W} \leftarrow \mathbf{W}_0 + \mathbf{A}\mathbf{B}^\top$
7: Replace $(\mathbf{W}_0, \mathbf{A}, \mathbf{B})$ with $\mathbf{W}$ in Θ.
8: **end for**
9: **end if**
10: **if** $t = D$ **then**
11: *// Deflate model back to low-rank*
12: **for** each full-rank weight matrix $\mathbf{W}$ in Θ **do**
13: Freeze the current weight $\mathbf{W}_f \leftarrow \mathbf{W}$.
14: Initialize new low-rank matrices $\mathbf{A}, \mathbf{B}$.
15: Replace $\mathbf{W}$ with $(\mathbf{W}_f, \mathbf{A}, \mathbf{B})$ in Θ.
16: **end for**
17: **end if**
18: Train the model on dataset $\mathcal{D}$ using Θ and η_t.
19: Update learning rate η_{t+1} according to the schedule.
20: **end for**

A Appendix

The appendix is structured as follows:

- Section A.1 presents a pseudocode of the proposed dynamic-rank training framework.
- Section A.2 summarizes experimental settings corresponding to all the experimental results reported in the main manuscript.
- Section A.3 describes how we inflate and deflate the model weights using Tucker and CP decompositions.
- Section A.4 describes how we adjust the rank of model weights at convolution layers.
- Section A.5 presents a singular value spectrum ratio comparison among different model rank adjustment settings.
- Section A.6 provides the results of additional ablation study on the impact of ϕ on model accuracy.
- Section A.7 summarizes potential limitations of our proposed dynamic-rank training framework.

We declare that an LLM was used to polish the writing. However, its purpose was solely to improve presentation quality and check grammar.

A.1 Algorithm

Algorithm 1 shows a pseudocode of the proposed dynamic-rank training framework. In this pseudocode, we assume the model is reparameterized using SVD. It is straightforward to replace SVD with other decomposition techniques. During E training epochs in total, if the epoch ID t becomes I, the model is inflated following the previously discussed reconstruction steps (line 5 $\sim$ 8). Likewise, if the epoch ID t becomes D, the model is deflated (line 10 $\sim$ 17). Other steps are the same as general neural network training process. Therefore, it does not have any dependencies on optimizers or model architectures.

A.2 Experimental Settings

CIFAR-10/CIFAR-100 datasets – We perform the typical image preprocessing used in many previous works (Lee et al., 2023) for CIFAR-10/100 datasets. 60,000 images with 50,000 images for train

Table 6: Hyper-parameter settings for experiments shown in Table 3. The ρ is the low-rank model's rank reduction ratio.

Dataset	Batch Size	Learning Rate	Epochs (E)	LR Decay	Inflate/Deflate (I, D)	Weight decay	ρ
CIFAR-10	128	0.1	150	100, 130	55, 120	$1e-4$	0.5
CIFAR-100			200	150, 180	80, 170	$5e-4$	
Tiny ImageNet	64		100	70, 90	30, 80	$5e-4$	

dataset and 10,000 images for validation dataset. Each image is padded by 4 pixels on every dimension and then randomly cropped to the original size. Then, we normalize and standardize the values for all individual pixels. Finally, with probability of 0.5 we randomly flip the image horizontally.

Tiny ImageNet dataset – For Tiny ImageNet with 200 classes, we augment the data samples during training as follows: aspect ratio adjustment [0.8, 1.25], random resizing [256, 384] pixels on shorter side, random cropping to 224×224 then resizing to 64×64, horizontal flipping with probability of 0.5, and HSV color augmentation (hue ± 36 degree, saturation/brightness [0.6, 1.4]). We normalize using ImageNet standard RGB values (mean [0.485, 0.456, 0.406], std [0.229, 0.224, 0.225]). For validation, we resize to 256 pixels on shorter side, center crop to 64×64, and apply the same normalization. We used 60,000 images with 50,000 images for train dataset and 10,000 images for validation dataset.

NLP datasets – We report performance on the GLUE development set following AdaLoRA (Zhang et al., 2023).

- **CoLA** (Warstadt et al., 2019): Judges if an English sentence is grammatically acceptable. (Train: 8.5k, Dev: 1k, Metric: Matthews Correlation Coefficient).
- **MNLI** (Williams et al., 2018): A 3-way classification task (entailment, neutral, contradiction) for sentence pairs across multiple genres. We use matched development set. (Train: 393k, Dev: 9.8k, Metric: Accuracy).
- **MRPC** (Dolan & Brockett, 2005): A binary classification task to determine if two sentences from online news are paraphrases. (Train: 3.7k, Dev: 408, Metric: Accuracy).
- **QNLI** (Rajpurkar et al., 2016): A binary classification task to identify if a context sentence contains the answer to a question. (Train: 105k, Dev: 5.4k, Metric: Accuracy).
- **QQP** (Iyer et al., 2017): A binary classification task to determine if two questions from Quora are semantically equivalent. (Train: 364k, Dev: 40k, Metric: Accuracy).
- **RTE** (Giampiccolo et al., 2007): A smaller, 2-way textual entailment classification task combining several datasets. (Train: 2.5k, Dev: 276, Metric: Accuracy).
- **SST-2** (Socher et al., 2013): A binary sentiment classification task on sentences from movie reviews. (Train: 67k, Dev: 872, Metric: Accuracy).
- **STS-B** (Cer et al., 2017): A regression task to predict a semantic similarity score (from 0 to 5) for sentence pairs. (Train: 5.7k, Dev: 1.5k, Metric: Pearson/Spearman Correlation).

Vision Experimental Settings – The Vision experiments detailed in Table 3 follow the configurations summarized in Table 6. We employed SGD optimizer with 0.9 momentum and conducted a grid search for learning rate, I, and D, executing each setting at least twice. The learning rate was tuned among 0.2, 0.1, 0.01. The I and D were first set to the midpoints of the high-noise and low-noise regimes, respectively. Then, each was finely tuned by grid search with a unit of 5. Table 6 presents the overall hyper-parameter settings we tuned.

NLP Experimental Settings – The NLP experiments presented in Table 4 are configured according to Table 7. We used AdamW optimizer with momentum 0.9, weight decay 1e-2, beta values (0.9, 0.999), sequence length 128, and 10% warm-up period of total steps. Through grid search performed at least twice per setting, we tuned learning rate among 1e-4, 5e-5, 2.5e-5, 1e-5, D from 2, 3, and λ among 0.5, 0.3, 0.1. Table 7 summarizes the highly tuned hyper-parameter settings.

Experimental Settings of Rank Recovery Analysis – The rank recovery experiments outlined in Table 5 follow the settings summarized in Table 8. We performed grid search for the algorithm-specific parameter, λ (regularizer coefficient), from 1e-3, 5e-4, 1e-4, 5e-5, 1e-5.

Table 7: Hyper-parameter settings for experiments shown in Table 4. The λ is the orthogonal regularizer coefficient in AdaLoRA (Zhang et al., 2023).

Dataset	Batch Size	Learning Rate	Epochs (E)	LoRA rank	α	Algorithm-specific parameter	Deflate (D)
COLA	16	$5e-5$	10	16	16	$\lambda = 0.5$	5
MNLI		$1e-5$	5			$\lambda = 0.1$	
MRPC		$1e-4$	5			$\lambda = 0.1$	2
QNLI		$1e-5$	5			$\lambda = 0.1$	
QQP		$1e-5$	5			$\lambda = 0.1$	
RTE		$5e-5$	10			$\lambda = 0.1$	5
SST-2		$1e-5$	5			$\lambda = 0.1$	2
SST-B		$1e-4$	5			$\lambda = 0.1$	

Table 8: Hyper-parameter settings for experiments shown in Table 5.

Method	Batch Size	Learning Rate	Epochs (E)	LR Decay	Algorithm-specific parameter	Inflate/Deflate (I, D)	ρ
SO	32	0.1	150	100, 130	$\lambda = 5e-5$	(55, 120)	0.5
DSO					$\lambda = 5e-5$		
SRIP$^+$					$\lambda = 5e-4$		

Table 9: Hyper-parameter settings for experiments shown in Table 10.

Dataset	Batch Size	Learning Rate	Epochs (E)	LR Decay	ϕ	Inflate/Deflate (I, D)	ρ
CIFAR-10	32	0.1	150	100, 130	0.1	92, 107	0.5
					0.3	75, 120	
					0.5	60, 135	
					0.7	45, 150	
					0.9	15, 150	
CIFAR-100	32	0.1	200	150, 180	0.1	140, 160	0.5
					0.3	120, 180	
					0.5	100, 200	
					0.7	60, 200	
					0.9	20, 200	

Experimental Settings of Ablation Study on Full-rank Epoch Budget – We conducted an ablation study examining how the number of full-rank epochs affects model accuracy and computational cost. Table 9 summarizes experimental settings corresponding to Table 10. We follow popularly used hyperparameter settings (e.g., a batch size of 32 and a learning rate of 0.1, etc.) and adjusted ϕ, I, and D. Given a fixed budget of ϕE full-rank epochs, we allocate half to the high-noise regime and half to the low-noise regime.

A.3 Model Inflation / Deflation with Various Decomposition Techniques

Our dynamic-rank training method is compatible with various decomposition techniques. In the main manuscript, we described how to inflate and deflate models using SVD as an example. Here, we explain how model weights are reparameterized using Tucker and CP decompositions. Let F denote the number of output channels, C the number of input channels, h the kernel height, w the kernel width, and k the reduced rank.

Model Inflation with Tucker decomposition – We define the model inflation process with Tucker decomposition (Kim et al., 2015) as follows. Given low-rank layer weights $\mathbf{A} \in \mathbb{R}^{1\times1\times C\times k}$, $\mathbf{Core} \in \mathbb{R}^{h\times w\times k\times k}$, $\mathbf{B} \in \mathbb{R}^{1\times1\times k\times F}$ and base parameter $\mathbf{W}_0 \in \mathbb{R}^{h\times w\times C\times F}$, the model is inflated such that $\mathbf{W} \leftarrow \mathbf{W}_0 + \mathbf{ACoreB}$.

Model Deflation with Tucker decomposition – Given a model weight $\mathbf{W}$, the model is deflated by attaching a low-rank adaptor path next to the original weight such that $\mathbf{W}_f + \mathbf{ACoreB} \leftarrow \mathbf{W}$, where $\mathbf{W}_f \in \mathbb{R}^{h\times w\times C\times F}$ is the given model weight matrix and $\mathbf{A} \in \mathbb{R}^{h\times w\times C\times F}$ and $\mathbf{A} \in \mathbb{R}^{1\times1\times C\times k}$, $\mathbf{Core} \in \mathbb{R}^{h\times w\times k\times k}$, and $\mathbf{B} \in \mathbb{R}^{1\times1\times k\times F}$ are low-rank model weights. The provided weight matrix is frozen as $\mathbf{W}_f$ and $\mathbf{A}$, $\mathbf{Core}$ and $\mathbf{B}$ are trained instead. One can initialize $\mathbf{A}$ and $\mathbf{B}$ using either random distributions or zero matrices.

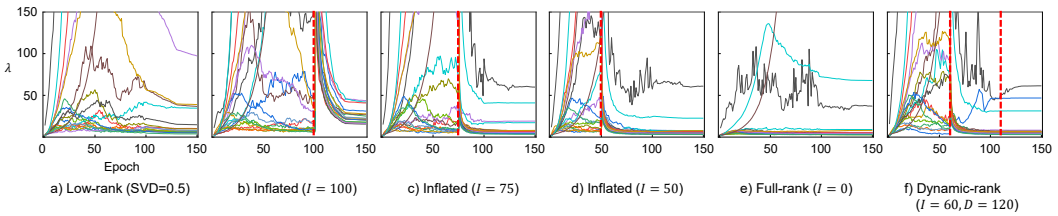

Figure 5: The singular value spectrum ratio λ comparison. The red dotted lines indicate the epoch where the rank of model weights are adjusted.

Model Inflation with CP decomposition – We define the model inflation process with CP decomposition (Lebedev et al., 2014) as follows. Given low-rank layer weights $\mathbf{A} \in \mathbb{R}^{1\times1\times C\times k}$, $\mathbf{C_1} \in \mathbb{R}^{h\times1\times k\times k}$, $\mathbf{C_2} \in \mathbb{R}^{1\times w\times k\times k}$, $\mathbf{B} \in \mathbb{R}^{1\times1\times k\times F}$ and base parameter $\mathbf{W}_0 \in \mathbb{R}^{h\times w\times C\times F}$, the model is inflated such that $\mathbf{W} \leftarrow \mathbf{W}_0 + \mathbf{AC_1C_2B}$.

Model Deflation with CP decomposition – Given a model weight $\mathbf{W}$, the model is deflated by attaching a low-rank adaptor path next to the original weight such that $\mathbf{W}_f + \mathbf{AC_1C_2B} \leftarrow \mathbf{W}$, where $\mathbf{W}_f \in \mathbb{R}^{m\times n}$ is the given model weight matrix and $\mathbf{A} \in \mathbb{R}^{1\times1\times C\times k}$, $\mathbf{C_1} \in \mathbb{R}^{h\times1\times k\times k}$, $\mathbf{C_2} \in \mathbb{R}^{1\times w\times k\times k}$, and $\mathbf{B} \in \mathbb{R}^{1\times1\times k\times F}$ are low-rank model weights. The provided weight matrix is frozen as $\mathbf{W}_f$ and $\mathbf{A}$, $\mathbf{C_1}$, $\mathbf{C_2}$ and $\mathbf{B}$ are trained instead. One can initialize $\mathbf{A}$ and $\mathbf{B}$ using either random distributions or zero matrices.

A.4 Rank Adjustment an Convolution Layers

Let F denote the number of output channels, C the number of input channels, h the kernel height, w the kernel width, and k the reduced rank. Note that we use a SVD-based approach as an example. Given low-rank convolution layer weights $\mathbf{A} \in \mathbb{R}^{h\times w\times C\times k}$, $\mathbf{B} \in \mathbb{R}^{1\times1\times k\times F}$ and base convolution layer weight $\mathbf{W_0} \in \mathbf{R}^{h\times w\times C\times F}$, convolution layer is inflated such that $\mathbf{W} \leftarrow \mathbf{W_0}+\mathbf{AB}$. Note that we assume $k < r \leq n$. If low-rank convolution follows the standard low-rank reparameterization method, the initial weight matrix $\mathbf{W_0}$ is set to zero matrix, i.e., $\mathbf{0_{h\times w\times C\times F}}$. Consequently, the maximum available rank is increased from k to n by training with inflated convolution layer. The deflation process follows the same steps, but in reverse order.

A.5 Singular Value Spectrum Ratio Comparison

To analyze the impact of interleaving full-rank epochs within low-rank training, we visualize the singular value spectrum ratio $\lambda^l, l \in [L]$ with various model inflation settings. Figure 5 presents the layer-wise λ curves of five different inflation settings. As I decreases, the full-rank epochs take up a large portion of the total epoch budget. We first observe that as I decreases, the λ^l values are more effectively suppressed, resulting in stable λ^l curves across most layers. For example, in the full-rank curves shown in Figure 5.e), all curves remain below $\lambda^l < 20$. When the model is inflated too late (e.g., $I = 100$), the curves stay relatively high, indicating that the model weights have lost their rank. As shown in Figure 5.f), when I is sufficiently small, the λ^l values are significantly reduced in most layers. Even after the model rank is deflated at $D = 120$, the λ^l values remain low until the end of training. This comparison provides clear insights into how to dynamically adjust the model rank during training to maximize the rank of the model weights.

Another intriguing observation is that the λ values at a few layers remain high regardless of rank adjustment. Notably, these layers consistently include the first and last layers across all experiments. Since their inputs or outputs remain fixed during training, their weights may rapidly fit to data patterns. If their rank can be suppressed, the model's overall capacity may be more effectively utilized, potentially achieving higher accuracy within the same epoch budget. We consider this an interesting direction for future research.

Table 10: Ablation study on how $\phi = (D - I)/E$ affects the performance of dynamic-rank training. Low-Rank SVD is used for all experiments. When $\phi = 0.0$, it becomes the conventional low-rank training. In contrast, when $\phi = 1.0$, it becomes full-rank training. The I and D are set to perform the same number of full-rank epochs in the high-noise and low-noise regimes.

Setting	Rank Ratio ρ	Comp.	CIFAR-10	CIFAR-100
$\phi = 0.1$	0.25	0.36	$90.54 \pm 0.3\%$	$76.29 \pm 0.1\%$
$\phi = 0.3$		0.51	$91.43 \pm 0.1\%$	$77.03 \pm 0.4\%$
$\phi = 0.5$		0.64	$92.01 \pm 0.3\%$	$78.61 \pm 0.2\%$
$\phi = 0.7$		0.79	$92.08 \pm 0.4\%$	$78.75 \pm 0.1\%$
$\phi = 0.9$		0.93	$92.14 \pm 0.2\%$	$78.63 \pm 0.3\%$
$\phi = 0.1$	0.5	0.61	$91.28 \pm 0.1\%$	$76.84 \pm 0.1\%$
$\phi = 0.3$		0.69	$91.57 \pm 0.1\%$	$77.51 \pm 0.2\%$
$\phi = 0.5$		0.78	$92.11 \pm 0.2\%$	$78.41 \pm 0.3\%$
$\phi = 0.7$		0.87	$92.15 \pm 0.2\%$	$78.39 \pm 0.1\%$
$\phi = 0.9$		0.96	$92.11 \pm 0.1\%$	$78.47 \pm 0.3\%$

A.6 Additional Ablation Study

In our empirical study, we found that dynamic-rank training performs well when D and I are set to allocate a similar number of full-rank epochs to both the high-noise and low-noise regimes. We now conduct a simple ablation study to examine how the number of full-rank epochs affects model accuracy. Table 10 presents the results. SVD-based low-rank training is evaluated on the CIFAR-10 and CIFAR-100 benchmarks. For convenience, we define $\phi = (D - I)/E$ as the ratio of full-rank epochs to the total training budget. When $\phi = 0.0$, it corresponds to conventional low-rank training; when $\phi = 1.0$, it becomes full-rank training. On both benchmarks, accuracy starts to drop when ϕ goes below 0.5, particularly in the range $\phi \in [0.3, 0.5]$. We therefore recommend starting with $\phi = 0.5$ and adjusting downward as needed.

A.7 Potential Limitations and Future Work

Potential Limitations – Our proposed method has a relatively more expensive computational cost compared to the conventional low-rank training methods. Since the number of trainable parameters increases during $D - I$ full-rank epochs, the overall cost is increased as shown in *Comp.* column in Table 3. However, the cost still remains substantially lower than that of the full-rank training, and we argue that the dynamic-rank training is a practical option for large-scale deep learning applications.

In addition, the extra hyperparameters, I and D, can introduce a non-trivial tuning overhead. However, Section 4.2 provides useful guidance on how to select good values for I and D based on learning rate decay schedules. In our empirical study, we find that I and D can be tuned easily by following our suggestions, leading to substantial accuracy improvement while maintaining low computational cost.

Future Work – We plan to extend our dynamic-rank training research to automate the inflation and deflation steps. Our study reveals that the inflation should be located at the end of the high-noise regime and the deflation should be as early as possible in the low-noise regime. If the noise scale can be quantified at run time, rather than explicitly mapping it to the learning rate decay schedule, appropriate timings for increasing and decreasing the model rank can be identified during training. We believe this will make the dynamic-rank training framework significantly more practical. Moreover, we consider harmonizing the dynamic-rank training framework with other existing compute-efficient training strategies as a promising direction for future work. Given the ever-increasing model sizes in deep learning applications (e.g., LLMs), developing neural network training strategies that balance accuracy and efficiency is a crucial research direction.