Neural Network Compression via Low-Rank Factorization

Jonah Bahr
Yale University
jonah.bahr@yale.edu

Abstract

Neural networks have become the standard architecture in machine learning. A typical deep learning solution involves an over-parameterized neural network, which is trained on large amounts of data. However, the recent exponential growth trend in model size highlights the need for effective compression methods that reduce the model footprint. In this work, we propose NN-SVD, a simple yet effective method that employs singular value decomposition to neural network layer weights to compress the model. We show that NN-SVD is theoretically robust under mild Lipschitz assumptions and demonstrates strong empirical performance on a small fully-connected model trained on the MNIST dataset.

1 Introduction

Singular value decomposition (SVD) is a linear dimension reduction technique that reveals various properties of the input matrix. Given any arbitrary real matrix $A \in \mathbb{R}^{m\times n}$, SVD produces three factor matrices: an orthonormal $U \in \mathbb{R}^{m\times m}$, a matrix of singular values $\Sigma \in \mathbb{R}^{m\times n}$, and an orthonormal $V \in \mathbb{R}^{n\times n}$, together which satisfy $A = U\Sigma V^\top$.

A well-known desirable property of SVD is that it provides a way to obtain the best rank-constrained approximation of the given matrix.

Theorem 1 (Eckart–Young–Mirsky [2])**.** *Let $A \in \mathbb{R}^{m\times n}$ be arbitrary. Suppose*

$$A = U\Sigma V^\top$$

is a singular value decomposition of A. Then the best rank k approximation of A under the Frobenius norm $\|\cdot\|_F$ is given by

$$A' = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top,$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ denote the i-th column of U and V, respectively.

Encouraged by Eckart–Young–Mirsky, in this work, we propose NN-SVD, a compression technique where we apply SVD low-rank factorization to neural network layer weights. Through theoretical and empirical analyses, we demonstrate that NN-SVD is provably robust and works well in practice. We also experiment with additional weight regularization techniques that impose explicit symmetry and implicit orthonormality constraints on the model.

2 Background

2.1 Model Compression

The objective of model compression is to decrease the size of a given model with minimal performance degradation. This is a challenging task since decreasing the model footprint by removing or shrinking its components typically adversely impacts its behavior. A commonly used model compression technique is weight pruning via sparsification, where entries of the weight matrix with values smaller than a given ϵ-threshold are set to zero [8]. Another popular approach is knowledge distillation [4], where the trained model assumes the role of a teacher, and a smaller student model is trained to match the final or intermediate outputs of the teacher model. For a survey of different model compression techniques, we refer readers to [9].

2.2 Low-Rank Training

Recent works [5, 12] suggest that the change in model weights that occur during language model training are low-rank. LoRA [5] proposes to train small low-rank adaptor neural networks instead of finetuning the entire model in full-rank. This significantly reduces computational requirements since training small adaptors are much cheaper than backpropagating on a large model. GaLore [12] extends this approach by using gradient low-rank projection, which allows the entire model to be trained efficiently with low-rank gradients.

In this work, we take the low-rank assumptions even further by explicitly imposing low-rank restrictions on the weight matrices of a neural network. Note that this is a tighter restriction than having a low-rank adaptor or using low-rank gradient projections since it is possible for the overall model to have full-rank weights in both LoRA and GaLore; in our method, we decompose the weights of the network such that they become low-rank.

3 Method

3.1 NN-SVD

Consider a neural network with N fully-connected layers,

$$f = f_N \circ f_{N-1} \circ \cdots \circ f_1.$$

Concretely, each layer f_i takes the form

$$f_i(\mathbf{x}) = \sigma(W_i \mathbf{x}),$$

where $\sigma(\cdot)$ is the activation function, and W_i is a weight matrix that parameterizes f_i. We omit the bias term for brevity, as it can be absorbed into the weight matrix W_i by introducing an auxiliary dimension to the input.

We also denote the sub-network within f from layers i through j inclusive, via

$$f_{i:j} \triangleq f_j \circ f_{j-1} \cdots \circ f_i.$$

Given a trained model f, a target layer index i, and the desired rank k, we compress f_i by applying SVD to its weight W_i, then choosing the top k largest singular values of Σ while truncating the columns of U and V to obtain the compressed weight $W_i' = U_{:k}\Sigma_{:k}V_{:k}^\top$, which parameterizes the compressed layer f_i'.

In practice, instead of storing $U_{:k}, \Sigma_{:k}$, and $V_{:k}$, we adopt an alternative representation with two factors

$$U' \triangleq U_{:k}\sqrt{\Sigma_{:k}}, V' \triangleq V_{:k}\sqrt{\Sigma_{:k}}^\top,$$

so that $U'V'^{\top} = W_i'$, thus requiring only $k(m+n)$ entries to be stored in memory. We can then define the compression rate.

Definition 1 (Compression Rate)**.** CR *is the ratio between the size of the full-rank layer weight and the compressed model weight after rank truncation to* k*.*

$$\mathrm{CR}(A, A_i) \triangleq \frac{mn}{k(m+n)}.$$

Corollary 1 (Maximal Rank for Compression)**.** *To realize memory savings so that* $\mathrm{CR}(W, W') \leq 1$*, we must have*

$$k \leq \frac{mn}{m-n}.$$

3.2 Approximation Error

Suppose we take f and apply a rank k approximation of f_i, resulting in

$$f' = f_{i+1:N} \circ f_i' \circ f_{1:i-1}.$$

We derive a concrete bound for the error that results from applying NN-SVD at the i-th layer, as well as a global error bound for the discrepancy between the final output of the original and compressed models after NN-SVD.

3.2.1 Layer Output Error

Proposition 1 (Layer Output Error Bound (Pre-activation))**.** *Suppose that the rank* k *approximation of* W*, given by* W'*, satisfies* [1]

$$\|W - W'\|_F \leq \epsilon.$$

Then given an input $\mathbf{x} \in \mathbb{R}^d$*, the output of the linear layer is bounded by*

$$\|W\mathbf{x} - W'\mathbf{x}\|_2 \leq \epsilon \|\mathbf{x}\|_2.$$

Proposition 1 tells us that given a suitable approximation W', the result of matrix multiplication does not yield a substantially different output as long as $\|\mathbf{x}\|_2$ is reasonably bounded. In practice, neural networks often accept normalized features that are mean-centered and also internally contain normalization layers, which ensures this assumption is met with high probability.

Notice that we have not considered the activation function $\sigma(\cdot)$. To include the activation function in our analysis, we consider a Lipschitz assumption on σ.

Definition 2 (L-Lipschitz)**.** *A function* $f : \mathbb{R}^m \to \mathbb{R}^n$ *is* L*-Lipschitz under the norm* $\|\cdot\|_p$ *if there exists a constant* L *such that* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^m$*,* f *satisfies*

$$\|f(\mathbf{x}) - f(\mathbf{y})\|_p \leq L \|\mathbf{x} - \mathbf{y}\|_p.$$

Assumption 1. *The activation function* $\sigma(\cdot)$ *is* L_σ*-Lipschitz under the* ℓ*-2 norm, i.e.,*

$$\|\sigma(\mathbf{x}) - \sigma(\mathbf{y})\|_2 \leq L_\sigma \|\mathbf{x} - \mathbf{y}\|_2.$$

This is a reasonable assumption, as popular activation functions, such as ReLU, softplus, and sigmoid have Lipschitz constants equal to 1. We also note that other common neural network layers such as dropout, batch normalization, and pooling all have simple and explicit Lipschitz constants [10, 3].

Now we can derive an error bound on the output of the fully-connected layer including the activation function.

[1]We drop the layer index subscript i for brevity.

Proposition 2 (Layer Output Error Bound (Post-activation))**.**

$$\|\sigma(W\mathbf{x}) - \sigma(W'\mathbf{x})\|_2 \leq \epsilon L_\sigma \|\mathbf{x}\|_2.$$

3.2.2 Model Output Error

We derive a global error bound that characterizes the difference between the final output of the original full-rank model and the compressed model. For this, we again invoke the Lipschitz property.

Lemma 1 (Lipschitzness of Linear Transformation)**.** *The linear transformation* $\mathbf{x} \to W\mathbf{x}$ *is* $\|W\|_F$*-Lipschitz:*

$$\|W\mathbf{x} - W\mathbf{y}\|_2 \leq \|W\|_F \|\mathbf{x} - \mathbf{y}\|_2.$$

Lemma 2 (Composition of Lipschitz Functions)**.** *Let* $g \triangleq g_M \circ g_{M-1} \circ \cdot \circ g_1$ *be a composition of* M *Lipschitz functions with constants* $L_1, L_2, \cdots, L_M$*. Then* g *is also Lipschitz, with constant*

$$L_g = \prod_{i=1}^{M} L_i.$$

This observation implies that any contiguous subset of the network is also Lipschitz, with the Lipschitz constant expressed in terms of L_σ and $\|W_i\|_F$.

Corollary 2. $\forall 1 \leq i < j \leq N, f_{i:j}$ *is Lipschitz:*

$$\|f_{i:j}(\mathbf{x}) - f_{i:j}(\mathbf{y})\|_2 \leq \left(L_\sigma^{j-i+1} \prod_{k=i}^{j} \|W_k\|_F \right) \|\mathbf{x} - \mathbf{y}\|_2.$$

We now derive the error bound on the final model output.

Proposition 3 (Model Output Error Bound)**.** *Let* $\mathbf{y} \triangleq f_{1:i-1}(\mathbf{x})$*. Then we have*

$$\|f(\mathbf{x}) - f'(\mathbf{x})\|_2 \leq \epsilon L_\sigma^{N-i} \|\mathbf{y}\| \prod_{j=i+1}^{N} \|W_j\|_F,$$

In summary, the difference between the output of the original full-rank model f and the compressed model f' is bounded in the ℓ-2 sense, with the bound depending on (1) ϵ, the goodness of the low rank approximation W'; (2) L_σ, the Lipschitz constant of the activation constant; (3) i, the layer at which the compression was applied, as well as N, the total number of layers of the network, (4) $\|W_j\|_F$, the norm of the weight matrices that parameterize f, and (5) $\|\mathbf{y}\|$, the output from the first $(i-1)$ layers of the neural network.

Although these terms could grow to be quite large, we hypothesize that the contribution from ϵ, when tuned correctly, will yield sufficiently accurate approximations so that the output of the compressed model will not deviate excessively from the original. In other words, NN-SVD will cause minimal performance degradation.

4 Result

4.1 Experiment

4.1.1 Setup

We evaluate our method on MNIST [7]. Our model is a simple 3-layer fully-connected neural network with 64 hidden dimensions, a $\mathrm{ReLU}(\cdot) \triangleq \max\{0, \cdot\}$ activation function, and dropout with $p = 0.5$. Given a single 28-by-28 pixel image, we flatten it to $\mathbb{R}^{784}$ and feed it into the neural network. We apply a fixed split to partition the dataset into 55,000 train and 5,000 test images. We train the model for 10 epochs with a batch size of 512 and a learning rate of 0.001 with the Adam optimizer [6]. All experiments were conducted on a single NVIDIA RTX 3090 GPU.

4.1.2 Low-Rank Training

Instead of low-rank compression, where we take an already trained model and apply NN-SVD, we consider a low-rank training scheme. This is inspired by recent works that suggest the change in model weights that occur during language model training are low-rank [5, 12]. Motivated by this analysis, we consider training rank-constrained neural networks from scratch. Since the model is explicitly trained with a rank constraint, given the same k value, a low-rank trained model should outperform a low-rank compressed model, which was not explicitly optimized under a low-rank setup. We include this baseline to establish an upper bound for NN-SVD.

4.1.3 Low-Rank Compression

We first train a baseline full-rank model. Then, we take the model checkpoint and apply NN-SVD on the network while varying the rank parameter k. Intuitively, the higher the k, the lower the compression rate CR and the closer the performance of the compressed model to the original; on the other hand, small values of k imply high compression, so we expect high performance degradation due to ϵ getting larger.

4.1.4 Orthonormal Regularization

Previous works [11, 1] have proposed that training models to have orthonormal weight vectors can have a positive impact on model performance. Inspired by these findings, we experiment with Double Soft Orthogonal (DSO) regularization [1].

Definition 3 (Double Soft Orthogonal Regularization)**.** *Let $W \in \mathbb{R}^{m \times n}$ denote the weight matrix and $\mathcal{L}$ denote any arbitrary loss objective function. Then the DSO-regularized loss is defined as*

$$\mathcal{L}_{\text{DSO}} \triangleq \mathcal{L} + \alpha \left(\left\| W^\top W - \mathbb{I}_n \right\|_F^2 + \left\| W W^\top - \mathbb{I}_m \right\|_F^2 \right)$$

Intuitively, we want $W^\top W$ and $WW^\top$ to approximate the identity matrix $\mathbb{I}$ to encourage both row and column-wise orthonormality. We experiment with $\alpha \in \{0.01, 0.001, 0.0001\}$ with rank constraint fixed at $k = 8$.

4.1.5 Symmetry Regularization

We consider an alternative regularization scheme, where we force the layer weights to be symmetric. There are many reasons why one might want to force layers to be symmetric: operations on symmetric matrices tend to be faster on modern hardware, and storing a symmetric matrix only requires half the memory as storing a non-symmetric matrix, which has positive implications for CR.

Instead of a soft regularization scheme, which might involve minimizing an auxiliary term such as $\left\| W - W^\top \right\|_F^2$, we explicitly parameterize the weight by inducing a decomposition $W = A^\top A$, and optimizing A. Then the output of this symmetrized layer can be written as $\sigma(A^\top A\mathbf{x})$.

4.2 Analysis

4.2.1 Rank

We find that the compression rank parameter k heavily impacts model performance. As shown in Figure 1, when the rank is constrained to small values, all models perform poorly, hovering around 20 to 40% accuracy. However, the accuracy scores increase as k gets larger, and at $k = 16$, all methods achieve parity with the full-rank baseline. This suggests that a large part of the model weights are in fact extraneous and that there exists a threshold at which a low-rank constraint is no longer detrimental to model performance.

We also find that NN-SVD, despite being entirely training-free, can match the performance of low-rank trained models, both with and without symmetry regularization. Initially, at $k = 1$, there is a noticeable gap between NN-SVD and low-ranked trained models, but the gap is slowly closed until reaching near-perfect parity at $k = 16$. This result not only corroborates our hypothesis that there exist extraneous components within the model's weight but also that NN-SVD is a viable model compression scheme to trim the model's footprint after full-rank training.

Figure 1: Rank vs Accuracy

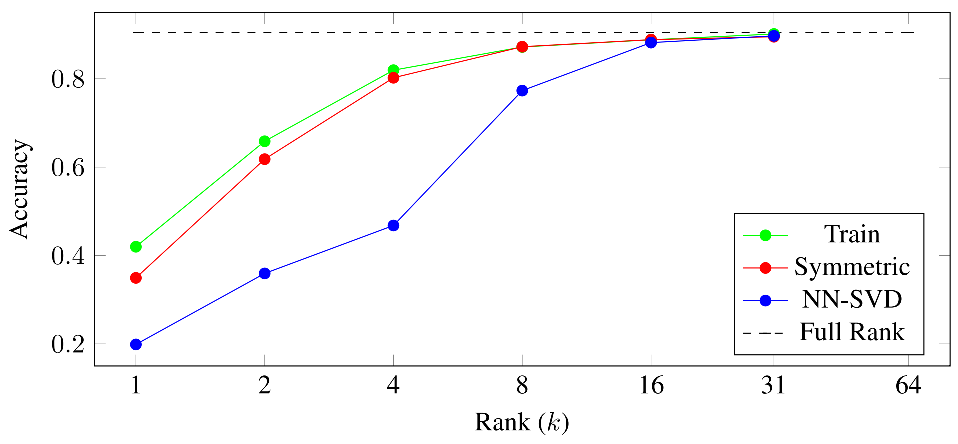
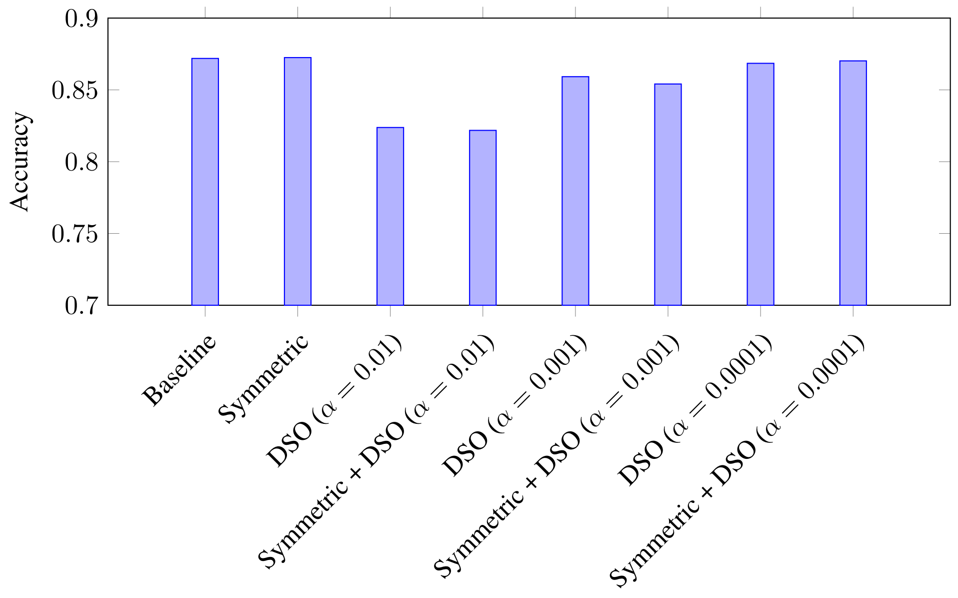


Figure 2: Regularization at $k = 8$



4.2.2 Regularization

Overall, we find that the two regularization techniques provide little to no benefit in model performance. As shown in Figure 1, symmetric models consistently lag behind the model without this constraint, although the gap closes at $k = 16$. However, we note that symmetric models require half the storage and memory required to hold a non-symmetric model. Therefore, symmetry regularization offers a way to achieve a trade-off between model performance and size, with both dimensions being a function of the compressed rank k. This offers users a way to flexibly design and compress their models to meet computational resource requirements.

On the other hand, we obtain negative results with orthonormal regularization via DSO. As shown in Figure 2, DSO-regularized models perform worse than the baseline, and in particular, decaying the DSO regularization weight α leads to strictly better performance. This trend continues even when DSO is combined with symmetry regularization. Since DSO requires additional backpropagation and gradient computation, we find that DSO increases the complexity of training while offering meager benefits to the user in our experiment setup.

5 Conclusion

In this work, we propose NN-SVD, a simple yet effective model compression technique that builds upon Eckart–Young–Mirsky. We show that, under mild assumptions on the norm of the input and Lipschitzness of the model, there exists an analytical upper bound for the ℓ-2 norm of the difference between the original model and the compressed model derived from NN-SVD. Through extensive empirical analyses, we also show that NN-SVD is able to closely match the performance of both the original model as well as low-rank trained models given a sufficient rank parameter k. We further demonstrate how low-rank trained models can be regularized via symmetry and orthonormality constraints. We hypothesize that the compressed rank k is intricately related to the manifold hypothesis: the higher the dimensionality of the data manifold, the more challenging the task of learning from the dataset, and hence the larger k required to retain model performance. We hope future works will further streamline this methodology by incorporating algorithms that automatically identify the optimal k value, thus saving users from the need to tune the compressed rank as an additional hyperparameter.

References

[1] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep cnns?, 2018.

[2] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[5] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[7] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits, 2005.

[8] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Neural Information Processing Systems*, 1989.

[9] James O' Neill. An overview of neural network compression, 2020.

[10] Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation, 2019.

[11] Yang Sui, Miao Yin, Yu Gong, Jinqi Xiao, Huy Phan, and Bo Yuan. Elrt: Efficient low-rank training for compact convolutional neural networks, 2024.

[12] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection, 2024.

A Proofs

A.1 Proposition 1

Proof. $\forall \mathbf{x} \in \mathbb{R}^d$, we have

$$\begin{aligned}\|W\mathbf{x} - W'\mathbf{x}\|_2 &= \|(W - W')\mathbf{x}\|_2 \\ &\leq \|W - W'\|_F \|\mathbf{x}\|_2 \\ &\leq \epsilon \|\mathbf{x}\|_2,\end{aligned}$$

where the first inequality follows from Cauchy-Schwarz. □

A.2 Proposition 2

Proof.

$$\begin{aligned}\|\sigma(W\mathbf{x}) - \sigma(W'\mathbf{x})\|_2 &\leq L_\sigma \|W\mathbf{x} - W'\mathbf{x}\|_2 \\ &\leq \epsilon L_\sigma \|\mathbf{x}\|_2,\end{aligned}$$

where the last inequality follows from Proposition 1. □

A.3 Lemma 1

Proof. This is the same calculation as Proposition 1.

$$\begin{aligned}\|W\mathbf{x} - W\mathbf{y}\|_2 &= \|W(\mathbf{x} - \mathbf{y})\|_2 \\ &\leq \|W\|_F \|\mathbf{x} - \mathbf{y}\|_2.\end{aligned}$$

□

A.4 Lemma 2

Suppose we have two Lipschitz functions $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^p$ with Lipschitz constants L_f and L_g respectively. We show that the composition $h = f \circ g$ is also Lipschitz with constant $L_h = L_f L_g$. The original statement can then be shown by induction.

Proof.

$$\begin{aligned}\|h(\mathbf{x}) - h(\mathbf{y})\| &= \|f(g(\mathbf{x})) - f(g(\mathbf{y}))\| \\ &\leq L_f \|g(\mathbf{x}) - g(\mathbf{y})\| \\ &\leq L_f L_g \|\mathbf{x} - \mathbf{y}\|.\end{aligned}$$

□

A.5 Corollary 2

Proof. We combine Assumption 1, Lemma 1, and Lemma 2. By Assumption 1 and Lemma 1, the Lipschitz constant of a single layer f_i is given by $L_\sigma \|W_i\|_F$. Then by Lemma 2, the Lipschitz constant of $f_{i:j}$ is given by

$$\prod_{k=i}^{j} L_\sigma \|W_k\|_F = L_\sigma^{j-i+1} \prod_{k=i}^{j} \|W_k\|_F.$$

□

A.6 Proposition 3

Proof. We use Corollary 2.

$$\begin{aligned}
\|f(\mathbf{x}) - f'(\mathbf{x})\|_2 &= \|(f_{i+1:N} \circ f_i \circ f_{1:i-1})(\mathbf{x}) - (f_{i+1:N} \circ f_i' \circ f_{1:i-1})(\mathbf{x})\|_2 \\
&= \|(f_{i+1:N} \circ f_i)(\mathbf{y}) - (f_{i+1:N} \circ f_i')(\mathbf{y})\|_2 \\
&= \|f_{i+1:N}(\sigma(W_i\mathbf{y})) - f_{i+1:N}(\sigma(W_i'\mathbf{y}))\|_2 \\
&\leq \left(L_\sigma^{N-i-1} \prod_{j=i+1}^{N} \|W_j\|_F \right) \|\sigma(W_i\mathbf{y}) - \sigma(W_i'\mathbf{y})\|_2 \\
&\leq \left(L_\sigma^{N-i} \prod_{j=i+1}^{N} \|W_j\|_F \right) \|W_i\mathbf{y} - W_i'\mathbf{y}\|_2 \\
&\leq \left(L_\sigma^{N-i} \prod_{j=i+1}^{N} \|W_j\|_F \right) \|W_i - W_i'\|_F \|\mathbf{y}\|_2 \\
&= \epsilon L_\sigma^{N-i} \|\mathbf{y}\| \prod_{j=i+1}^{N} \|W_j\|_F.
\end{aligned}$$

□