# Implicit Bias in Matrix Factorization and its Explicit Realization in a New Architecture

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Gradient descent for matrix factorization is known to exhibit an implicit bias toward approximately low-rank solutions. While existing theories often assume the boundedness of iterates, empirically the bias persists even with unbounded sequences. We thus hypothesize that implicit bias is driven by divergent dynamics markedly different from the convergent dynamics for data fitting. Using this perspective, we introduce a new factorization model: $X \approx UDV^\top$, where $U$ and $V$ are constrained within norm balls, while $D$ is a diagonal factor allowing the model to span the entire search space. Our experiments reveal that this model exhibits a strong implicit bias regardless of initialization and step size, yielding truly (rather than approximately) low-rank solutions. Furthermore, drawing parallels between matrix factorization and neural networks, we propose a novel neural network model featuring constrained layers and diagonal components. This model achieves strong performance across various regression and classification tasks while finding low-rank solutions, resulting in efficient and lightweight networks.

## 1 Introduction

The Burer–Monteiro (BM) factorization (Burer & Monteiro, 2003) is a classical technique for obtaining low-rank solutions in optimization. One can view it as a simple neural network that uses a single layer of hidden neurons under linear activation. Indeed, given the factorization $X = UV^T$ where $U \in \mathbb{R}^{d\times r}$ and $V \in \mathbb{R}^{c\times r}$, one can view $U$ and $V$ as the weights of the first and second layers, and $r$ as the number of hidden neurons. But despite the similarity suggested by this view, there is a clear distinction between BM factorization and neural networks in how the rank $r$ is chosen. In BM, $r$ is typically chosen to be small, close to the rank of the desired solution. Neural networks, on the other hand, often succeed even in overparametrized settings where $r$ is large.

Recent findings of implicit regularization in matrix factorization narrow the gap between these two perspectives. For instance, Gunasekar et al. (2017) demonstrate that gradient descent (with certain parameter selection) on BM factorization tends to converge toward approximately low-rank solutions even when $r = d$. Based on this observation, they conjecture that *"with small enough step sizes and initialization close enough to the origin, gradient descent on full-dimensional factorization converges to the minimum nuclear norm solution."*

In a follow-up work, Razin & Cohen (2020) present a counterexample demonstrating that implicit regularization in BM factorization *cannot be explained by minimal nuclear norm*, or in fact any norm. Specifically, they show that there are instances where the gradient method applied to BM factorization yields a diverging sequence, and all norms thus grow toward infinity. Intriguingly, despite this divergence, they found that the rank of the estimate decreases toward its minimum.

Although this phenomenon might seem surprising initially, it is not uncommon for diverging sequences to follow a structured path. A prime example is the Power Method, the fundamental algorithm

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

for finding the largest eigenvalue and eigenvector pair of a matrix. Starting from a random initial point $x_0$, the Power Method iteratively updates the estimate by multiplying it with the matrix. This process amplifies the component of the vector that aligns with the direction of the dominant eigenvector more than the other components, progressively leading $x_k$ to align with this eigenvector. In practical implementations, $x_k$ is scaled after each iteration to avoid numerical issues from divergence.

This perspective underpins our approach. Specifically, our key insight is that the implicit regularization in BM factorization (and neural networks) is driven by divergent dynamical behavior. This is markedly different from the standard (convergent) optimization dynamics helping with the data fitting. In this context, we hypothesize that these forces do not merely coexist but actively compete, influencing model behavior and performance in fundamentally conflicting ways. Our main goal in the development of this paper is to devise an approach that unravels these competing forces.

### 1.1 Overview of main contributions

- **A novel formulation for matrix factorization**. We model $X = UDV^\top$, where $U$ and $V$ are constrained within Frobenius norm balls. Projection onto this ball results in a scaling step similar to the Power Method. The middle term $D$ is a diagonal matrix that allows the model to explore the entire search space despite $U$ and $V$ being bounded.[1]

  Through extensive empirics we demonstrate that the gradient method applied to the proposed formulation exhibits a pronounced implicit bias toward low-rank solutions. We compare our formulation against standard BM factorization with two unconstrained factors. Specifically, we investigate key factors such as step size and initialization, which prior work suggests might be contributing to implicit bias. We find that our factorization approach largely obviates the need to rely on these conditions: it consistently finds ***truly (rather than approximately) low-rank solutions*** across a wide range of initializations and step-sizes in our experiments. We believe these findings should be of broader interest to research on implicit bias.
- **A novel architecture**. Motivated by the strong bias for low-rank solutions of the proposed factorization, we subsequently extend it to deep neural networks. We do so by adding constrained layers and diagonal components. We show that this constrained model performs on par with, or even better than, the standard architecture across various regression and classification tasks. Importantly, our approach exhibits bias towards low-rank solutions, resulting in a natural pruning procedure that delivers compact, lightweight networks without compromising performance.

### 1.2 Related Work

**Burer-Monteiro factorization.** BM factorization was proposed for solving semidefinite programs (Burer & Monteiro, 2003, 2005) and has been recognized for its efficiency in addressing low-rank optimization problems (Boumal et al., 2016; Park et al., 2018). Building on the connections between matrix factorization and training problems for two-layer neural networks, BM models have served as foundational building blocks for understanding implicit bias and developing theoretical insights.

**Implicit regularization.** One promising line of research that aims to explain the successful generalization abilities of neural networks is that of 'implicit regularization' induced by the optimization methods and architectures (Neyshabur et al., 2014, 2017; Neyshabur, 2017). Several studies explore matrix factorization to investigate implicit bias (Gunasekar et al., 2017; Arora et al., 2018; Razin & Cohen, 2020; Belabbas, 2020; Li et al., 2021). Much of the existing work focuses on gradient flow dynamics in the limit of infinitesimal learning rates. Exceptionally, Gidel et al. (2019) examine discrete gradient dynamics in two-layer linear neural networks, showing that the dynamics progressively learn solutions of reduced-rank regression with a gradually increasing rank.

**Constrained neural networks.** Regularizers are frequently used in neural network training to prevent overfitting and improve generalization, or to achieve structural benefits such as sparse and compact network architectures (Scardapane et al., 2017). However, it is conventional to apply these regularizers as penalty functions in the objective rather than constraints. This approach is likely favored due to the ease of implementation, as pre-built functions are readily available in common neural network packages. Regularization in the form of constraints appears to be rare in neural network training. One notable exception is in the context of neural network training with the Frank-Wolfe algorithm (Pokutta et al., 2020; Zimmer et al., 2022; Macdonald et al., 2022). Recently, Pethick et al. (2025)

[1]The reader may notice a "syntactic" similarity with SVD; except using vastly simpler Frobenius norm constraints on $U$ and $V$ instead of orthogonality.

revealed parallels between Frank-Wolfe on constrained networks and algorithms that post-process update steps, such as Muon (Jordan et al., 2024), which achieves state-of-the-art results on nanoGPT by orthogonalizing the update directions before applying them.

**Pruning.** Neural networks are overparameterized, which can enhance generalization and avoid poor local minima. But such models then suffer from excessive memory and computational demands, making them less efficient for deployment in real-world applications (Chang et al., 2021). Pruning reduces the number of parameters, resulting in more compact and efficient models that are easier to deploy. A comprehensive review on pruning is beyond the scope of this paper due to space limitations and the diversity of approaches. We refer to (Reed, 1993; Blalock et al., 2020; Cheng et al., 2024) and the references therein for detailed reviews. Pruning by singular value thresholding has recently shown promising results, particularly in natural language processing (Chen et al., 2021), and is often used along with various enhancements such as importance weights and data whitening for effective compression of large language models (Hsu et al., 2022; Yuan et al., 2023; Wang et al., 2024).

## 2 Matrix Factorization with a Diagonal Component

Consider *matrix sensing*, a problem where we seek to recover a *positive semidefinite* (PSD) matrix $X \in \mathbb{S}_+^{d\times d}$ from a set of linear measurements $b = \mathcal{A}(X) \in \mathbb{R}^n$. We define $\mathcal{A} : \mathbb{R}^{d\times d} \to \mathbb{R}^n$ through symmetric measurement matrices $A_1, \dots, A_n \in \mathbb{S}^{d\times d}$, such that $\mathcal{A}(X) = [\langle A_1, X\rangle \ \cdots \ \langle A_n, X\rangle]^\top$ and $\mathcal{A}^\top y = \sum_{i=1}^n y_i A_i$. We particularly focus on the data-scarce setting where $n \ll d^2$. A notable example here matrix completion, where one completes a matrix $X$ given a subset of its entries. This problem is inherently under-determined; but successful recovery is possible if $X$ is low-rank (Candes & Recht, 2012). We focus on recovering a PSD matrix for simplicity; this is without loss of generality, as the general case can be be easily reformulated as a PSD matrix sensing problem (Park et al., 2017).

The problem described above can be cast as the following rank-constrained optimization problem:

$$\min_{X\in\mathbb{S}_+^{d\times d}} \quad f(X) := \tfrac{1}{2}\|\mathcal{A}(X) - b\|_2^2 \quad \text{subj. to} \quad \operatorname{rank}(X) \le r. \tag{1}$$

Although rank-constrained matrix optimization problems are typically NP-hard, various methods have been developed to provide practical approximations. One prominent approach is BM factorization, which reparametrizes the decision variable $X$ as $UU^\top$, where the factor $U \in \mathbb{R}^{d\times r}$, and $r$ is a positive integer that controls the rank of the resulting product. Problem (1) can then be reformulated as:

$$\min_{U\in\mathbb{R}^{d\times r}} \quad \tfrac{1}{2}\|\mathcal{A}(UU^\top) - b\|_2^2. \tag{2}$$

Despite the fact that finding the global minimum of (2) remains challenging, a local solution can be approximated using gradient descent (Lee et al., 2016). Initializing at $U_0 \in \mathbb{R}^{d\times r}$, perform:

$$U_{k+1} = U_k - \eta\nabla_U f(U_k U_k^\top), \tag{3}$$

where $\eta > 0$ is the step-size, and the gradient is computed as $\nabla_U f(UU^\top) = 2\nabla f(UU^\top)U$.

Selecting the factorization rank $r$ is a critical decision. A small $r$ may lead to spurious local minima, resulting in inaccurate outcomes (Waldspurger & Waters, 2020). Conversely, a large $r$ might weaken rank regularization, rendering the problem underdetermined. Conventional wisdom in BM factorization suggests finding a moderate compromise between these two extremes. However, a key observation in (Gunasekar et al., 2017) is that the gradient method applied to (2) exhibits a tendency towards approximately low-rank solutions even when $r = d$. Below, we restate their conjecture:

**Conjecture in (Gunasekar et al., 2017).** Suppose gradient flow (*i.e.,* gradient descent with an infinitesimally small step-size) is initialized at a *full-rank matrix arbitrarily close to the origin.* If the limit of the gradient flow, $X_{\text{GF}} = UU^\top$, exists and is a global optimum of (1) with $\mathcal{A}(X_{\text{GF}}) = b$, then $X_{\text{GF}}$ is the minimal nuclear-norm solution to (1).

### 2.1 The Proposed Factorization

We propose reparameterizing $X = UDU^\top$, where $U \in \mathbb{R}^{d\times r}$ is constrained to have a bounded norm, and $D \in \mathbb{R}^{r\times r}$ is a non-negative diagonal matrix:

$$\min_{\substack{U\in\mathbb{R}^{d\times r}\\ D\in\mathbb{R}^{r\times r}}} \frac{1}{2}\|\mathcal{A}(UDU^\top) - b\|_2^2 \quad \text{s.t.} \quad \|U\|_F \le \alpha, \ \ D_{ii} \ge 0, \ \ D_{ij} = 0, \ \ \forall i \text{ and } \forall j \ne i, \tag{4}$$

where $\alpha > 0$ is a model parameter. When the problem is well-scaled, for instance through basic preprocessing with data normalization, we found that $\alpha = 1$ is a reasonable choice.

Placing in multiple factors and with constraints, we perform projected-gradient updates on $U$ and $D$ with step-size $\eta > 0$:

$$\begin{aligned} U_{k+1} &= \Pi_U \left( U_k - \eta \nabla_U f(U_k D_k U_k^\top) \right) \\ D_{k+1} &= \Pi_D \left( D_k - \eta \nabla_D f(U_k D_k U_k^\top) \right), \end{aligned} \tag{5}$$

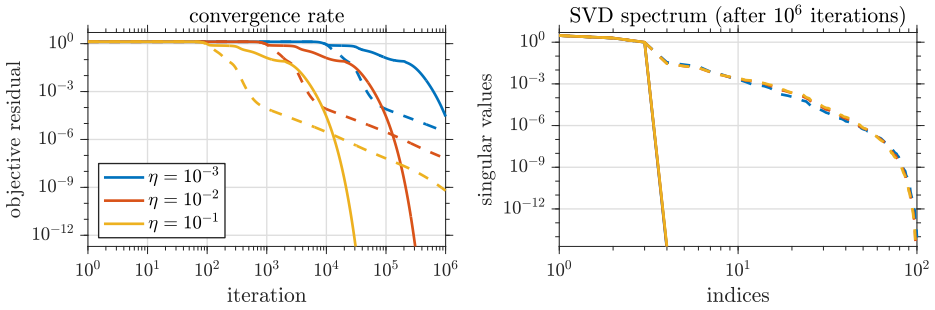
where $\Pi_U$ and $\Pi_D$ are projections for the constraints in (4); while the gradients are

$$\nabla_U f(UDU^\top) = 2\nabla f(UDU^\top)UD \quad \text{and} \quad \nabla_D f(UDU^\top) = U^\top \nabla f(UDU^\top)U.$$

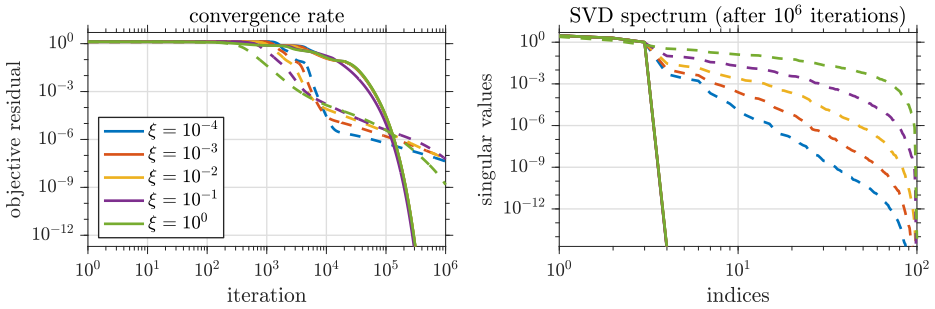
## 2.2 Numerical Experiments on Matrix Factorization

We present numerical experiments comparing the empirical performance of the proposed approach with the classical BM factorization. Specifically, we examine the impact of initialization and step-size on the singular value spectrum of the resulting solution. We set up a synthetic matrix completion problem to recover a PSD matrix $X_\natural = U_\natural U_\natural^\top$, where the entries of $U_\natural \in \mathbb{R}^{100\times 3}$ are drawn independently from $N(0,1)$. We randomly sample $n = 900$ entries of $X_\natural$ and store them in the vector $b \in \mathbb{R}^n$. The goal is to recover $X_\natural$ from $b$ by solving problems (2) and (4). For initialization, we generate $U_0 \in \mathbb{R}^{d\times d}$ with entries drawn independently from $N(0,1)$; we rescale $U_0$ to have Frobenius norm $\xi > 0$ (we investigate the impact of $\xi$). We initialize $D_0 = I$.

The results are shown in Figure 1. First, we examine the impact of step-size. To this end, we fix $\xi = 10^{-2}$ and test different values of $\eta$. In the left panel, we plot the objective residual as a function of iterations. As expected, we observe that a smaller step-size slows down convergence. In the right panel, we plot the singular value spectrum of the results attained after $10^6$ iterations. We observe no direct connection between step-size and implicit bias in BM factorization.




Impact of **step-size** ($\eta$), in **noiseless** setting, with fixed initialization.




Impact of **initial distance to origin** ($\xi$), in **noiseless** setting, with fixed step-size.

Figure 1: Impact of step-size and initialization on implicit bias. **Solid lines represent our UDU factorization**, while **dashed lines denote the classical BM factorization**. [*Left*] Objective residual vs. iterations. [*Right*] Singular value spectrum after $10^6$ iterations. In all cases, UDU produces truly low-rank solutions, whereas the classical approach results in approximate low-rank structures.

Next, we investigate the impact of initialization. We fix the step-size at $\eta = 10^{-2}$ and evaluate the effect of varying $\xi$. We observe a correlation between the implicit bias of the BM factorization and $\xi$,

which determines the initial distance from the origin. Initializing closer to the origin in the classical BM factorization yields solutions with a faster spectral decay. Notably, the UDU factorization demonstrates a strong implicit bias toward truly low-rank solutions, regardless of the choice of $\eta$ or $\xi$.

We provide additional experiments in the Appendices. Specifically, Appendix A.1 considers the matrix completion problem with noisy measurements. The results remain consistent with the noiseless case: the UDU model exhibits an implicit bias toward truly low-rank solutions, while the classical BM factorization yields approximately low-rank solutions. Additionally, we present numerical experiments on a matrix sensing problem arising in phase retrieval image recovery in Appendix A.2. As before, the UDU framework consistently promotes low-rank solutions, and this structural bias significantly enhances the quality of the recovered image.

### 2.3 Theoretical Insights into the Inner Workings and Implicit Bias

A fixed-point analysis of the proposed method provides valuable insights into its inner workings. Define the update variables before projection as $\bar{U} = U - 2\eta\nabla f(X)UD$ and $\bar{D} = D - \eta U^\top \nabla f(X)U$, with $X = UDU^\top$. Suppose $(U, D)$ is a fixed point of the algorithm in (5). Then, the following hold:

Let $u_j$ denote the $j^{\text{th}}$ column of $U$ and $\lambda_j$ the $j^{\text{th}}$ diagonal entry of $D$.

$(a)$ If $\|\bar{U}\| \leq \alpha$, then $\nabla f(X)u_j\lambda_j = 0$ for all $j$,

$(b)$ If $\|\bar{U}\| > \alpha$, then there exists some $\beta > 0$ such that $\nabla f(X)u_j\lambda_j = -\beta u_j$ for all $j$.

At this point, it may seem that choosing a small value of $\alpha$ could promote a fixed point where the columns of $U$ align with the negative eigenvectors of $\nabla f(X)$. However, as we will see from the analysis of $D$, there are no valid fixed points that satisfy $\|\bar{U}\| > \alpha$, since

$(c)$ If $\lambda_j = 0$, then $u_j^\top \nabla f(X)u_j \geq 0$, , while $(d)$ If $\lambda_j > 0$, then $u_j^\top \nabla f(X)u_j = 0$.

Suppose $\|\bar{U}\| > \alpha$. Then, $(b)$ implies that if $\lambda_j > 0$, then $u_j$ must be an eigenvector of $\nabla f(X)$ corresponding to a negative eigenvalue; and if $\lambda_j = 0$, then $u_j$ must also be zero. However, the first statement contradicts $(d)$, while the second statement agrees with $(c)$ only if $u_j = 0$. Since these conditions must hold for all $j$, it follows that $U = 0$. This, in turn, implies that $\bar{U} = 0$, which contradicts the initial assumption that $\|\bar{U}\| > \alpha$, hence there are no fixed points satisfying $\|\bar{U}\| > \alpha$.

Considering $(a)$, observe that the fixed point characterization obtained here coincides with the fixed points of the BM factorization after the change of variables from $U$ to $UD^{1/2}$. Thus, incorporating constraints on $U$ and adding the factor $D$ does not introduce any new fixed points for $X$.

Interestingly, this fixed point analysis also provides insight into the low-rank bias of the algorithm. In particular, when $U$ tends to grow and $\|\bar{U}\|$ exceeds $\alpha$, the algorithm appears to temporarily favor directions where the columns of $U$ align with the negative eigenvectors of $\nabla f(X)$. To see this more concretely, we can express the update rules in terms of the columns of $\bar{U}$ and the diagonal entries of $\bar{D}$ as $\bar{u}_j = u_j - 2\eta\nabla f(X)u_j\lambda_i$ and $\bar{\lambda}_j = \lambda_j - \eta u_j^\top \nabla f(X)u_j$. These expressions show that both $\bar{u}_j$ and $\bar{\lambda}_j$ tend to grow in the directions aligned with the negative eigenvectors of $\nabla f(X)$. However, from statement $(d)$, we know that there is no fixed point with $\lambda_j > 0$ unless $u_j^\top \nabla f(X)u_j = 0$. This suggests that: (i) either the algorithm might push $u_j$ towards zero, which could happen only through the projection steps if another column $\bar{u}_{j'}$ exhibits a faster growth; or (ii) $X$ should evolve such that $f(X)$ is minimized in the direction of $u_j$, effectively moving towards a point where $\nabla f(X)u_j = 0$.

The analysis presented here is a simplified perspective aimed at gaining insight. In reality, the alignment or shrinking of the columns of $U$ and the minimization of $f(X)$ along specific directions reflected in these columns occur simultaneously and interact in a complex manner. Nevertheless, we can clearly observe these effects in our numerical experiments. In Appendix A.3, we present the evolution of the column norms of $U$ and the diagonal entries of $D$ over the iterations in our matrix completion experiment. Our results show that initially, a few specific columns of $U$ grow, pushing all other columns numerically to zero. Once $f(X)$ is effectively minimized with respect to these initial columns, some other columns are identified and start to grow. Eventually, the algorithm converges to a low-rank solution, where the factorization $UDU^\top$ is rank-revealing since only a few columns of $U$ are nonzero. We further observe that these nonzero columns are orthogonal, effectively demonstrating how the algorithm's specific preference to align $u_j$ with the negative eigenvectors of $\nabla f(X)$ along the path implicitly induces a structured solution.

## 3 Feedforward Neural Networks with Diagonal Hidden Layers

This section extends our approach to neural networks. Consider a dataset comprising $n$ data points $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^d \times \mathbb{R}^c$. We first define a three-layer neural network defined as

$$\phi(\mathbf{x}) := \sum_{j=1}^{m} \mathbf{v}_j w_j \mathbf{u}_j^\top \mathbf{x} \;\; \approx \;\; \mathbf{y}. \tag{6}$$

The first and third layers are fully connected, and the middle is a diagonal layer, as illustrated in Figure 2. Drawing parallels between our matrix factorization model in (4) and neural network training, we impose Euclidean norm constraints on the weights of the fully connected layers. Under these conditions, the training problem can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{u}_j, w_j, \mathbf{v}_j} \quad & \frac{1}{2n} \sum_{i=1}^{n} \| \sum_{j=1}^{m} \mathbf{v}_j \, w_j \, \mathbf{u}_j^\top \mathbf{x}_i - \mathbf{y}_i \|_2^2 \\ \text{subj.to} \quad & \sum_{j=1}^{m} \|\mathbf{u}_j\|_2^2 \le 1, \quad \sum_{j=1}^{m} \|\mathbf{v}_j\|_2^2 \le 1, \;\; \text{and} \;\; w_j \ge 0; \quad \text{for all } \; j = 1, \ldots, m. \end{aligned} \tag{7}$$

The norm constraints in our training problem can be interpreted as a stronger form of weight decay, one of the most commonly used regularization techniques in neural networks, which lends further justification to our formulation. We refer to this neural network structure as UDV.

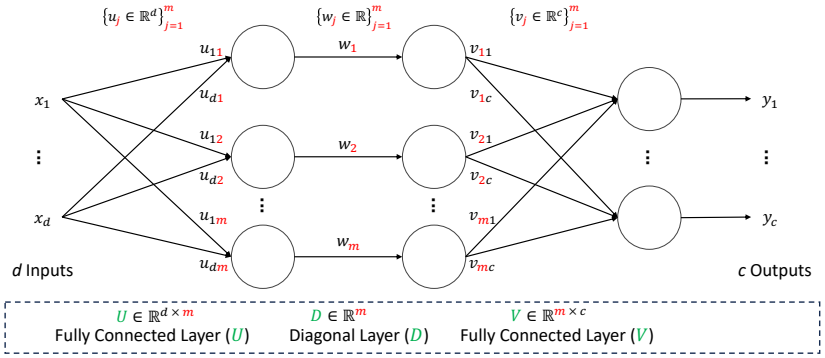



Figure 2: UDV structure. The weights in diagonal layer $D$ are denoted as $w_j$.

### 3.1 Numerical Experiments on Neural Networks

In this section, we test the proposed UDV framework on regression and classification tasks, comparing it with fully connected two-layer neural networks (denoted as UV in the subsequent text) using both linear and ReLU activation functions. This comparison is fair in terms of computational cost, as the cost incurred by the diagonal layer —which can also be viewed as a parameterized linear activation function— is negligible. We observe a strong empirical bias toward low-rank solutions in all our experiments. We also present a proof-of-concept use case of this strong bias, combined with an SVD-based pruning strategy, to produce compact networks.

#### 3.1.1 Implementation Details

**Computing environment.** All classification tasks were conducted on an NVIDIA A100 GPU with four cores of the AMD Epyc 7742 processor, while regression tasks were conducted on a single core of an Intel Xeon Gold 6132 processor. We used Python 3.9.5 and PyTorch 2.0.1.

**Datasets.** We used two datasets for the regression tasks: House Prices - Advanced Regression Techniques (HPART) (Anna Montoya, 2016) and New York City Taxi Trip Duration (NYCTTD) (Risdal, 2017). We allocated 80% of the data for the training and reserved the remaining 20% for validation. Following (Huang, 2003), we set the number of hidden neurons in the diagonal layer $m = \texttt{round}\big(\sqrt{(c+2)d} + 2\sqrt{d/(c+2)}\big)$. This results in a network structure ($d$-$m$-$c$) of 79-26-1 for HPART, and 12-10-1 for NYCTTD.

For classification tasks, we used the normalized MNIST dataset (LeCun et al., 2010). We applied transfer learning by replacing the classifier layers of three advanced neural networks –MaxViT-T (Tu

Table 1: Model performance using different models. M, E, and R represent the transferred models MaxVit-T, EfficientNet-B0, and RegNetX-32GF, respectively.

| Tasks | Regression (Test Loss) | | Classification (Test Accuracy) | | |
|---|---|---|---|---|---|
| Dataset | HPART | NYCTTD | MNIST | | |
| UDV | $1.304 \times 10^{-3}$<br>Adam: $10^{-3}$ | $5.248 \times 10^{-6}$<br>NAdam: $10^{-4}$ | M: 99.67%<br>MBGDM: $10^{-2}$ | E: 99.63%<br>MBGDM: $10^{-1}$ | R: 99.74%<br>MBGDM: $10^{-2}$ |
| UV | $1.333 \times 10^{-3}$<br>Adam: $10^{-3}$ | $5.251 \times 10^{-6}$<br>Adam: $10^{-3}$ | M: 99.69%<br>MBGDM: $10^{-2}$ | E: 99.60%<br>Adam: $10^{-3}$ | R: 99.66%<br>MBGD: $10^{0}$ |
| UV-ReLU | $1.167 \times 10^{-3}$<br>Adam: $10^{-3}$ | $5.323 \times 10^{-6}$<br>NAdam: $10^{-3}$ | M: 99.68%<br>NAdam: $10^{-4}$ | E: 99.68%<br>MBGDM: $10^{-1}$ | R: 99.73%<br>MBGD: $10^{0}$ |

et al., 2022), EfficientNet-B0 (Tan & Le, 2019), and RegNetX-32GF (Radosavovic et al., 2020)– with UDV, while using pre-trained weights from ImageNet-1K (Deng et al., 2009). Specifically, we retained all layers up to the first fully connected layer of the classifier and replaced the subsequent layers. The number of hidden neurons in the diagonal layer was set to as $m = \texttt{floor}(\frac{2}{3}d)$. This results in a UDV network structure ($d$-$m$-$c$) of 512-341-10 for MaxViT-T, 1280-853-10 for EfficientNet-B0, and 2520-1680-10 for RegNetX-32GF.

**Loss function.** We used mean squared error for regression and cross-entropy loss for classification.

**Optimization methods.** We tested the results using four different optimization algorithms for training: Adam (Kingma & Ba, 2014), Mini-Batch Gradient Descent (MBGD) (LeCun et al., 2002), NAdam (Dozat, 2016), and Mini-Batch Gradient Descent with Momentum (MBGDM) (Sutskever et al., 2013). For classification, we used different batch sizes for different models: 128 for MaxViT-T and RegNetX-32GF, and 384 for EfficientNet-B0.

We tuned the step size for all models and optimization algorithms: For regression tasks, we tested step sizes $10^{-4}$, $10^{-3}$, $10^{-2}$, $10^{-1}$, 1, 2, 3. Larger step sizes (1, 2, 3) were often excluded for the $UV$ model due to divergence. For classification we tested LRs $10^{-6}$, $10^{-5}$, $10^{-4}$, $10^{-3}$, $10^{-2}$, $10^{-1}$, 1 with Adam and NAdam; and we tested $10^{-3}$, $10^{-2}$, $10^{-1}$, 1, 2, 3, 5 with MBGD and MBGDM.

**Training Procedure.** UV and UDV models were initialized identically, with the diagonal elements of $D$ initialized using Kaiming Uniform Initialization, consistent with the default initialization for fully connected layers in PyTorch. The models were trained for 200 epochs on HPART, 50 epochs on NYCTTD, and 70 epochs on MNIST; and results were averaged over 1000 random seeds for HPART, 100 for NYCTTD, and 1 for MNIST to ensure robustness. The validation loss, used as a generalization metric in regression, was averaged over the final 20 epochs for the HPART dataset and the final 5 epochs for the NYCTTD dataset. Similarly, validation accuracy for classification tasks was averaged over the last 5 epochs to ensure stability in the reported values.

### 3.1.2 Low-rank Bias in Neural Network Training

Table 1 presents the validation loss (for regression) or validation accuracy (for classification) of the UDV model compared to the classical UV model with linear and ReLU activation functions. For each configuration (dataset and model architecture), the results are obtained by selecting the best algorithm and learning rate pair. Moreover, Figure 3 illustrates the singular value spectrum of the solutions corresponding to each entry in these tables. We focus on the singular values from the $U$ and $UD$ layers, as they generate the primary data representation, while omitting the $V$ layer, which serves as the feature selection layer and is a tall matrix by definition, given that $c \ll m$ in most cases. Collectively, these results show that the UDV framework achieves competitive prediction accuracy while exhibiting a strong implicit bias toward low-rank solutions, as indicated by the faster decay in the singular value spectrum.

### 3.1.3 Reducing Network Size with SVD-based Pruning

Efficient and lightweight feed-forward layers are crucial for real-world applications. For instance, the Apple Intelligence Foundation Models (Gunter et al., 2024) recently reported that pruning hidden dimensions in feed-forward layers yields the most significant gains in their foundation models.

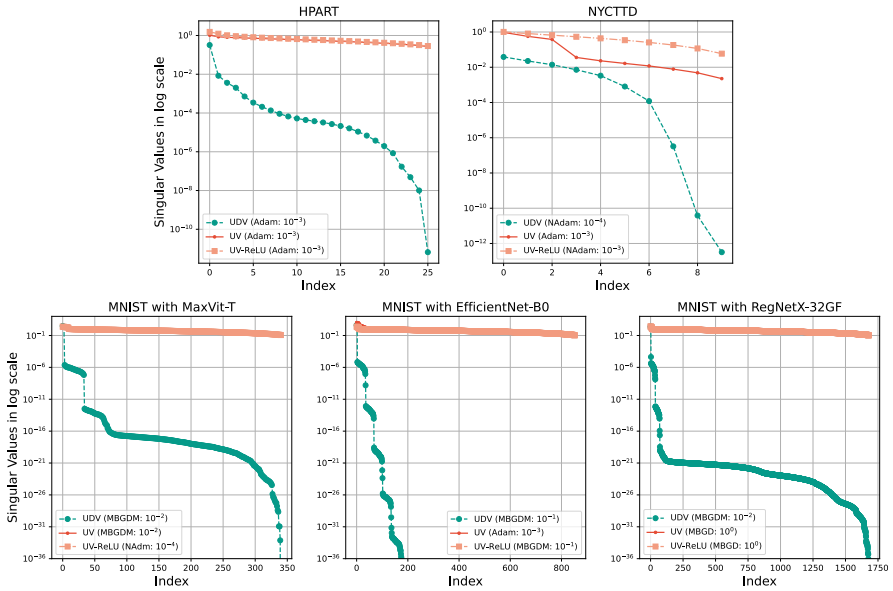



Figure 3: Singular value spectrum corresponding to the solutions reported in Table 1.

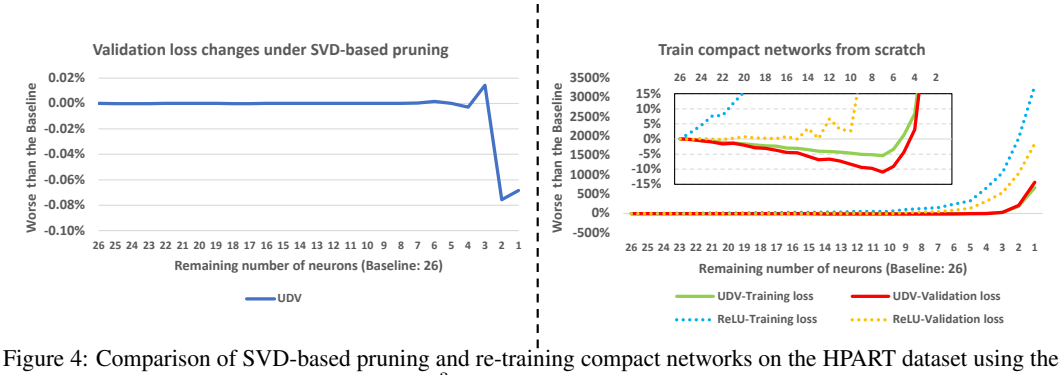



Figure 4: Comparison of SVD-based pruning and re-training compact networks on the HPART dataset using the NAdam algorithm with a learning rate of $10^{-3}$. Negative percentages indicate improvements over the baseline. SVD-based pruning demonstrates that the UDV leads to a compact model without performance degradation, while retraining shows that the UDV achieves better generalization in a compact model.

Building on this insight, we leverage the inherent low-rank bias of the UDV architecture through an SVD-based pruning strategy to produce compact networks without sacrificing performance.

A low-rank solution was observed when applying SVD to $UD$ layers:

$$UD = \mathtt{U}\mathtt{S}\mathtt{V}^\top, \quad \mathtt{U} \in \mathbb{R}^{d\times m}, \quad \mathtt{S} \in \mathbb{R}^{m\times m}, \quad \mathtt{V}^\top \in \mathbb{R}^{m\times m}. \tag{8}$$

By dropping small singular values in $\mathtt{S}$, these matrices can be truncated to $\bar{\mathtt{U}} \in \mathbb{R}^{d\times r}$, $\bar{\mathtt{S}} \in \mathbb{R}^{r\times r}$ and $\bar{\mathtt{V}}^\top \in \mathbb{R}^{r\times m}$, where $0 < r < m$. Consequently, $(m - r)$ neurons can be pruned, and new weight matrices are assigned:

$$\bar{U} = \bar{\mathtt{U}} \in \mathbb{R}^{d\times r}, \quad \bar{D} = \bar{\mathtt{S}} \in \mathbb{R}^{r\times r}, \quad \bar{V} = \bar{\mathtt{V}}^T V \in \mathbb{R}^{r\times c}. \tag{9}$$

We applied this pruning strategy on the models from Table 1. The left part of Figure 4 presents an example comparing the generalization capability of pruned models. For comparison, we created compact models by training from scratch with a reduced number of neurons $m$ in the hidden layer. The performance change for these models is shown in the right panel of Figure 4. Although our pruned networks derived from UDV demonstrate that models with significantly fewer parameters can still achieve strong generalization, these compressed architectures are often more challenging to optimize directly within the reduced space, consistent with prior findings in the literature (Arora et al., 2018; Chang et al., 2021). We omit the results for retraining with the UV model, as they show similar trends to UDV in this context, though UDV generally exhibits superior generalization.

### 3.1.4 Further Details and Discussions

Our findings in experiments on neural networks align with the results observed in matrix factorization. A key distinction, however, was the use of different optimization algorithms, including stochastic gradients and momentum steps, in neural network experiments. Despite these differences, the UDV architecture consistently demonstrated a strong bias toward low-rank solutions. Additional experiments and details can be found in the Appendices, with key results summarized below:

- In the early stages of this research, we explored four variants of UDV, each differing slightly in their constraints. We selected the version presented in (7), as it generally exhibits the most pronounced decay in the singular value spectrum. For completeness, details of the other three variants are provided in Appendix B.1.
- Appendix B.2 provides additional results for the experiment described in Section 3.1.2. Specifically, we present results analogous to those in Table 1 and Figure 3, but focusing exclusively on the MBGDM algorithm. These results exhibit similar trends, reinforcing consistency across different methods. Additionally, comprehensive performance comparisons across all algorithms and models are provided in Tables SM2 to SM5 in the Appendices.
- Additional results on SVD-based pruning are provided in Appendix B.3. We demonstrate that the UDV framework consistently achieves low-rank solutions across various problem configurations. Furthermore, we analyze the effect of learning rate on the singular value spectrum, similar to the analysis in Figure 1, but applied to neural network experiments. This analysis confirms that the UDV framework produces low-rank solutions across a broad range of learning rates.
- Appendix B.4 extends the UDV framework by incorporating ReLU activation. Preliminary experiments with the UDV-ReLU model indicate that it also tends to yield low-rank solutions, similar to those observed in the original UDV framework.
- Prior work on implicit bias in neural networks suggests that increasing depth enhances the tendency toward low-rank solutions (Arora et al., 2019; Feng et al., 2022), raising the question of whether the pronounced bias in the UDV framework is just a consequence of adding a diagonal layer. To investigate this, we conducted experiments comparing the UDV model to fully connected three-layer networks, as detailed in Appendix B.5. Additionally, we included a UDV model without constraints in these comparisons. The results indicate that this bias cannot be attributed solely to depth, highlighting the critical role of explicit constraints.
- Appendix B.6 compares the spectral decay of the UDV network to that induced by classical weight decay regularization in two- and three-layer networks. While weight decay promotes singular value decay, it can not reproduce the strong decay observed in the UDV model.
- Appendix B.7 presents a toy example demonstrating the application of the UDV structure within the LoRA framework to fine-tune a pre-trained LLaMA-2 model on a causal language modeling task. The pruning results highlight the potential of the proposed UDV block to replace linear layers in a wide range of models, demonstrating promising performance under compression.

## 4 Conclusions

We proposed a new matrix factorization framework, inspired by the observation that implicit bias is driven by dynamics that are distinct from those leading to convergence of the objective function. This framework constrains the factors within Euclidean norm balls and introduces a middle diagonal factor to ensure the search space is not restricted. Numerical experiments demonstrate that this approach significantly strengthens the low-rank bias in the solution.

To explore the broader applicability of our findings, we designed an analogous neural network architecture with three layers, constraining the fully connected layers and adding a diagonal hidden layer, referred to as UDV. Extensive experiments show that the proposed UDV architecture achieves competitive performance compared to standard fully connected networks, while inducing a structured solution with a strong bias toward low-rank representations. Additionally, we explored the utility of this low-rank structure by applying an SVD-based pruning strategy, illustrating how it can be leveraged to construct compact networks that are more efficient for downstream tasks.

The proposed model exhibits reduced rank regression behavior, where the training process gradually increases the model rank. While this promotes a low-rank structure, it can also slow convergence. Although we provide some theoretical insights, developing a more complete theory and designing algorithms that fully exploit the model's regularization capabilities, especially for large-scale problems, remain important directions for future work.

# References


DataCanary Anna Montoya. House prices - advanced regression techniques, 2016. URL https://kaggle.com/competitions/house-prices-advanced-regression-techniques.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pp. 254–263. PMLR, 2018.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

Mohamed Ali Belabbas. On implicit regularization: Morse functions and applications to matrix factorization. *arXiv preprint arXiv:2001.04264*, 2020.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 29, 2016.

Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical programming*, 95(2):329–357, 2003.

Samuel Burer and Renato DC Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical programming*, 103(3):427–444, 2005.

Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

Emmanuel J Candes, Thomas Strohmer, and Vladislav Voroninski. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Mathematics*, 66(8):1241–1274, 2013.

Xiangyu Chang, Yingcong Li, Samet Oymak, and Christos Thrampoulidis. Provable benefits of overparameterization in model compression: From double descent to pruning neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6974–6983, 2021.

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Drone: Data-aware low-rank compression for large nlp models. *Advances in neural information processing systems*, 34:29321–29334, 2021.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Timothy Dozat. Incorporating Nesterov momentum into Adam. 2016.

Ebrahim Elgazar. Pixel art, 2024. URL https://www.kaggle.com/datasets/ebrahimelgazar/pixel-art/.

Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank diminishing in deep neural networks. *Advances in Neural Information Processing Systems*, 35: 33054–33065, 2022.

Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. *Advances in neural information processing systems*, 30, 2017.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. Apple Intelligence foundation language models. *arXiv preprint arXiv:2407.21075*, 2024.

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.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Guang-Bin Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE transactions on neural networks*, 14(2):274–281, 2003.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 2002.

Yann LeCun, Corinna Cortes, and Christopher J. Burges. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. In *International Conference on Learning Representations*, 2021.

Jan Macdonald, Mathieu E Besançon, and Sebastian Pokutta. Interpretable neural networks with Frank-Wolfe: Sparse relevance maps and relevance orderings. In *International Conference on Machine Learning*, pp. 14699–14716. PMLR, 2022.

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

Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

Dohyung Park, Anastasios Kyrillidis, Constantine Carmanis, and Sujay Sanghavi. Non-square matrix sensing without spurious local minima via the Burer-Monteiro approach. In *Artificial Intelligence and Statistics*, pp. 65–74. PMLR, 2017.

Dohyung Park, Anastasios Kyrillidis, Constantine Caramanis, and Sujay Sanghavi. Finding low-rank solutions via nonconvex matrix factorization, efficiently and provably. *SIAM Journal on Imaging Sciences*, 11(4):2165–2204, 2018.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

Sebastian Pokutta, Christoph Spiegel, and Max Zimmer. Deep neural network training with Frank-Wolfe. *arXiv preprint arXiv:2010.07243*, 2020.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces, 2020.

Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. *Advances in neural information processing systems*, 33:21174–21187, 2020.

Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

Meg Risdal. New York City taxi trip duration, 2017. URL https://kaggle.com/competitions/nyc-taxi-trip-duration.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-axis vision transformer. In *European conference on computer vision*, pp. 459–479. Springer, 2022.

Irene Waldspurger and Alden Waters. Rank optimality for the burer–monteiro factorization. *SIAM journal on Optimization*, 30(3):2577–2602, 2020.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.

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

Max Zimmer, Christoph Spiegel, and Sebastian Pokutta. Compression-aware training of neural networks using Frank-Wolfe. *arXiv preprint arXiv:2205.11921*, 2022.

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Main claims in abstract and introduction section can reflect the contributions which are also mentioned in the introduction section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in the conclusions section, along with future directions.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Theory assumptions and proofs are provided in the section "Matrix Factorization with a Diagonal Component".

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The experiments are fully reproducible. The datasets we used are properly cited and publicly available, and the code with sufficient instructions is included in the supplemental material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The datasets we used are properly cited and publicly available, and the code with sufficient instructions (including the pre-process of datasets) is included in the supplemental material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified the above details in the main paper (see section "Numerical experiments on matrix factorization" and the section "Numerical experiments on neural networks"), and the submitted code includes full details with instructions.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Results, especially the neural networks with UDV, were averaged sufficient random initialization seeds. We introduced how we get robust results in the subsection "Implementation details"

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specify the computer resources in the subsection "Implementation details", where there is a paragraph beginning with "Computing environment ...". Wall time is 7 days for CPU works and 3 days for GPU works.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research respects the NeurIPS code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: Justification: This research is related to neural network optimization and does not have direct societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: There is no high risk of misuse associated with this research.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and models based on transfer learning were appropriately cited in this research.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not refer to the proposed methods as new assets since the datasets and basic models are existing assets and are properly cited.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLM is only used for polishing language.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.