

---

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

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

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

## 15 1 Introduction

16 The Burer–Monteiro (BM) factorization (Burer & Monteiro, 2003) is a classical technique for  
17 obtaining low-rank solutions in optimization. One can view it as a simple neural network that uses a  
18 single layer of hidden neurons under linear activation. Indeed, given the factorization  $X = UV^T$   
19 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,  
20 and  $r$  as the number of hidden neurons. But despite the similarity suggested by this view, there is a  
21 clear distinction between BM factorization and neural networks in how the rank  $r$  is chosen. In BM,  
22  $r$  is typically chosen to be small, close to the rank of the desired solution. Neural networks, on the  
23 other hand, often succeed even in overparametrized settings where  $r$  is large.

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

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

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

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.<sup>1</sup>

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)

<sup>1</sup>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  $\mathcal{A} : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}^n$  through symmetric measurement matrices  $A_1, \dots, A_n \in \mathbb{S}^{d \times d}$ , such that  $\mathcal{A}(X) = [\langle A_1, X \rangle \dots \langle A_n, X \rangle]^\top$  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 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:

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

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:

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

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:

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

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:

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

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  $\eta > 0$ :

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

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.

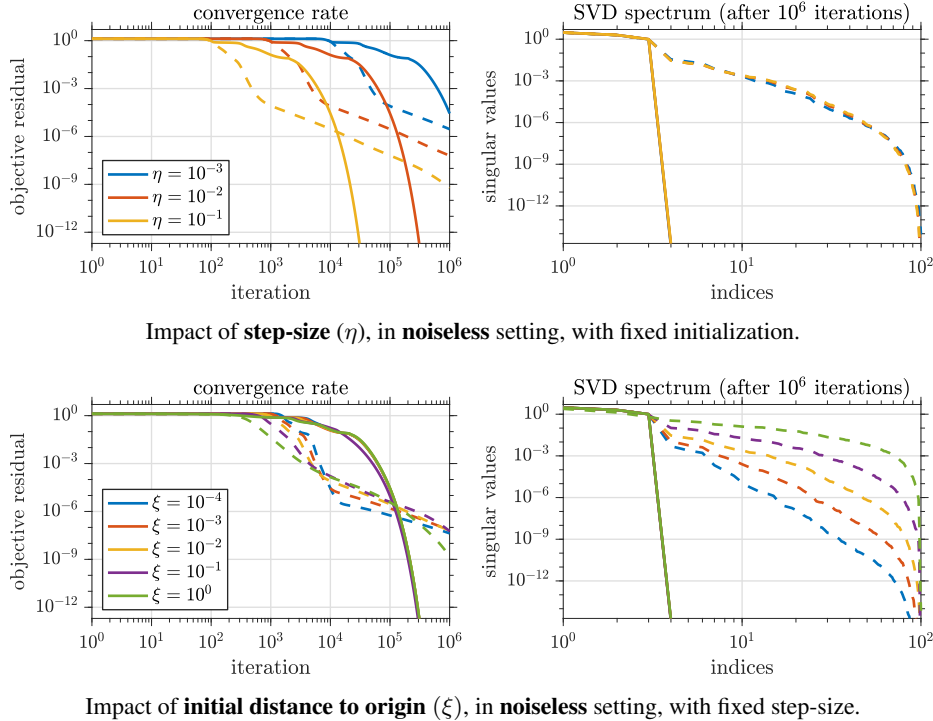


Figure 1: Impact of step-size and initialization on implicit bias. **Solid lines represent our UDU factorization, while dashed lines denote the classical BM factorization.** [Left] Objective residual vs. iterations. [Right] Singular value spectrum after  $10^6$  iterations. In all cases, UDU produces 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  $\bar{u}_j = u_j - 2\eta \nabla f(X)u_j\lambda_j$  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

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

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:

$$\begin{aligned} \min_{\mathbf{u}_j, w_j, \mathbf{v}_j} \quad & \frac{1}{2n} \sum_{i=1}^n \left\| \sum_{j=1}^m \mathbf{v}_j w_j \mathbf{u}_j^\top \mathbf{x}_i - \mathbf{y}_i \right\|_2^2 \\ \text{subj.to} \quad & \sum_{j=1}^m \|\mathbf{u}_j\|_2^2 \leq 1, \quad \sum_{j=1}^m \|\mathbf{v}_j\|_2^2 \leq 1, \quad \text{and} \quad w_j \geq 0; \quad \text{for all } j = 1, \dots, m. \end{aligned} \quad (7)$$

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.

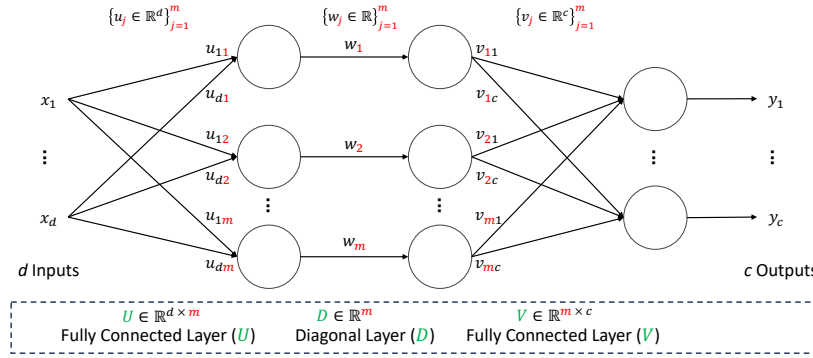


Figure 2: UDV structure. The weights in diagonal layer  $D$  are denoted as  $w_j$ .

#### 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  $m = \text{round}(\sqrt{(c+2)d} + 2\sqrt{d/(c+2)})$ . 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	$1.304 \times 10^{-3}$ Adam: $10^{-3}$	$5.248 \times 10^{-6}$ NAdam: $10^{-4}$	M: 99.67% MBGDM: $10^{-2}$	E: 99.63% MBGDM: $10^{-1}$	R: 99.74% MBGDM: $10^{-2}$
UV	$1.333 \times 10^{-3}$ Adam: $10^{-3}$	$5.251 \times 10^{-6}$ Adam: $10^{-3}$	M: 99.69% MBGDM: $10^{-2}$	E: 99.60% Adam: $10^{-3}$	R: 99.66% MBGD: $10^0$
UV-ReLU	$1.167 \times 10^{-3}$ Adam: $10^{-3}$	$5.323 \times 10^{-6}$ NAdam: $10^{-3}$	M: 99.68% NAdam: $10^{-4}$	E: 99.68% MBGDM: $10^{-1}$	R: 99.73% 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  $m = \text{floor}(\frac{2}{3}d)$ . 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.

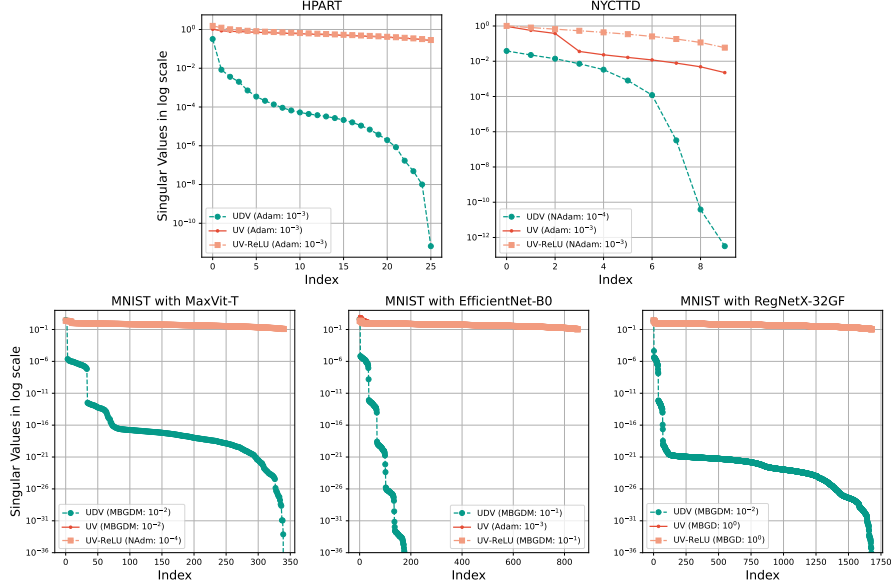


Figure 3: Singular value spectrum corresponding to the solutions reported in Table 1.

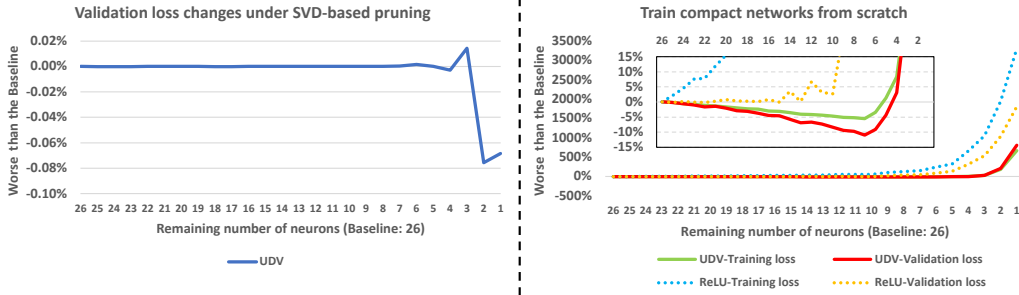


Figure 4: Comparison of SVD-based pruning and re-training compact networks on the HPART dataset using the NAdam algorithm with a learning rate of  $10^{-3}$ . Negative percentages indicate improvements over the baseline. SVD-based pruning demonstrates that the UDV leads to a compact model without performance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

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

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

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

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

$$\bar{U} = \bar{U} \in \mathbb{R}^{d \times r}, \quad \bar{D} = \bar{S} \in \mathbb{R}^{r \times r}, \quad \bar{V} = \bar{V}^\top V \in \mathbb{R}^{r \times c}. \quad (9)$$

279 We applied this pruning strategy on the models from Table 1. The left part of Figure 4 presents an  
 280 example comparing the generalization capability of pruned models. For comparison, we created  
 281 compact models by training from scratch with a reduced number of neurons  $m$  in the hidden layer.  
 282 The performance change for these models is shown in the right panel of Figure 4. Although our  
 283 pruned networks derived from UDV demonstrate that models with significantly fewer parameters  
 284 can still achieve strong generalization, these compressed architectures are often more challenging  
 285 to optimize directly within the reduced space, consistent with prior findings in the literature (Arora  
 286 et al., 2018; Chang et al., 2021). We omit the results for retraining with the UV model, as they show  
 287 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.
- Armin Askari, Geoffrey Negiar, Rajiv Sambharya, and Laurent El Ghaoui. Lifted neural networks. *arXiv preprint arXiv:1805.01532*, 2018.
- Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. In *International Conference on Learning Representations*, 2019.
- Burak Bartan and Mert Pilanci. Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time. *arXiv preprint arXiv:2101.02429*, 2021.
- Mohamed Ali Belabbas. On implicit regularization: Morse functions and applications to matrix factorization. *arXiv preprint arXiv:2001.04264*, 2020.
- Srinadh Bhojanapalli, Anastasios Kyrillidis, and Sujay Sanghavi. Dropping convexity for faster semi-definite optimization. In *Conference on Learning Theory*, pp. 530–582. PMLR, 2016.
- Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Gutttag. 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.
- Nicolas Boumal, Vladislav Voroninski, and Afonso S Bandeira. Deterministic guarantees for Burer-Monteiro factorizations of smooth semidefinite programs. *Communications on Pure and Applied Mathematics*, 73(3):581–608, 2020.
- 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.
- Diego Cifuentes. On the burer-monteiro method for general semidefinite programs. *Optimization Letters*, 15(6):2299–2309, 2021.

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

393 Timothy Dozat. Incorporating Nesterov momentum into Adam. 2016.

394 Ebrahim Elgazar. Pixel art, 2024. URL <https://www.kaggle.com/datasets/ebrahimelgazar/pixel-art/>.  
395

396 Tolga Ergen and Mert Pilanci. Implicit convex regularizers of CNN architectures: Convex optimiza-  
397 tion of two-and three-layer networks in polynomial time. In *International Conference on Learning  
398 Representations*, 2020.

399 Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank  
400 diminishing in deep neural networks. *Advances in Neural Information Processing Systems*, 35:  
401 33054–33065, 2022.

402 Jonathan Fiat, Eran Malach, and Shai Shalev-Shwartz. Decoupling gating from linearity. *arXiv  
403 preprint arXiv:1906.05032*, 2019.

404 Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient  
405 dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 32,  
406 2019.

407 Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

408 Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A  
409 survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

410 Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro.  
411 Implicit regularization in matrix factorization. *Advances in neural information processing systems*,  
412 30, 2017.

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

416 Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for  
417 efficient neural network. *Advances in neural information processing systems*, 28, 2015.

418 Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon.  
419 *Advances in neural information processing systems*, 5, 1992.

420 Ya-Ping Hsieh, Yu-Chun Kao, Rabeeh Karimi Mahabadi, Alp Yurtsever, Anastasios Kyrillidis, and  
421 Volkan Cevher. A non-Euclidean gradient descent framework for non-convex matrix factorization.  
422 *IEEE Transactions on Signal Processing*, 66(22):5917–5926, 2018.

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

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

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

429 Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks  
430 with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

431 Steven A Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600,  
432 1989.

433 Zhipeng Jia, Xingyi Huang, I Eric, Chao Chang, and Yan Xu. Constrained deep weak supervision for  
434 histopathology image segmentation. *IEEE transactions on medical imaging*, 36(11):2376–2388,  
435 2017.

436 Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy  
437 Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL <https://kellerjordan.github.io/posts/muon/>.  
438

439 Hoel Kervadec, Jose Dolz, Meng Tang, Eric Granger, Yuri Boykov, and Ismail Ben Ayed. Constrained-  
440 CNN losses for weakly supervised segmentation. *Medical image analysis*, 54:88–99, 2019.

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

443 Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A  
444 whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

445 Thien Le and Stefanie Jegelka. Training invariances and the low-rank phenomenon: beyond linear  
446 networks. In *International Conference on Learning Representations*, 2022.

447 Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-  
448 up convolutional neural networks using fine-tuned CP-decomposition. In *International Conference*  
449 *on Learning Representations*, 2015.

450 Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard,  
451 and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances*  
452 *in neural information processing systems*, 2, 1989a.

453 Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information*  
454 *processing systems*, 2, 1989b.

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

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

459 Ching-pei Lee, Ling Liang, Tianyun Tang, and Kim-Chuan Toh. Accelerating nuclear-norm reg-  
460 ularized low-rank matrix optimization through Burer-Monteiro decomposition. *arXiv preprint*  
461 *arXiv:2204.14067*, 2022.

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

464 Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. SNIP: Single-shot network pruning  
465 based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

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

469 Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson.  
470 Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on*  
471 *Scientific Computing*, 43(6):B1105–B1132, 2021.

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

475 Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep  
476 networks: Promises and limitations. *arXiv preprint arXiv:1706.02025*, 2017.

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

479 Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a  
480 network via relevance assessment. *Advances in neural information processing systems*, 1, 1988.

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

482 Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the  
483 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.

Rahul S Patel, Sharad Bhartiya, and Ravindra D Gudi. Physics constrained learning in neural network based modeling. *IFAC-PapersOnLine*, 55(7):79–85, 2022.

Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE international conference on computer vision*, pp. 1796–1804, 2015.

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.

Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pp. 7695–7705. PMLR, 2020.

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

Karthik Prakhya, Tolga Birdal, and Alp Yurtsever. Convex formulations for training two-layer ReLU neural networks. *arXiv preprint arXiv:2410.22311*, 2024.

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

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

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>.

Mehmet Fatih Sahin, Ahmet Alacaoglu, Fabian Latorre, Volkan Cevher, et al. An inexact augmented Lagrangian framework for nonconvex optimization with nonlinear constraints. *Advances in Neural Information Processing Systems*, 32, 2019.

Arda Sahiner, Tolga Ergen, John M Pauly, and Mert Pilanci. Vector-output ReLU neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms. In *International Conference on Learning Representations*, 2021.

Arda Sahiner, Tolga Ergen, Batu Ozturkler, John M Pauly, Morteza Mardani, and Mert Pilanci. Scaling convex neural networks with Burer-Monteiro factorization. In *The Twelfth International Conference on Learning Representations*, 2023.

Tara N Sainath, Brian Kingsbury, Vikas Sindhvani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6655–6659. IEEE, 2013.

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



533 Ruoyu Sun and Zhi-Quan Luo. Guaranteed matrix completion via non-convex factorization. *IEEE*  
534 *Transactions on Information Theory*, 62(11):6535–6579, 2016.

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

538 Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank  
539 factorization for deep neural network compression. *Neurocomputing*, 398:185–196, 2020.

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

542 Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit regularization towards rank minimization in  
543 ReLU networks. In *International Conference on Algorithmic Learning Theory*, pp. 1429–1459.  
544 PMLR, 2023.

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

548 Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao  
549 Li. MaxViT: Multi-axis vision transformer. In *European conference on computer vision*, pp.  
550 459–479. Springer, 2022.

551 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz  
552 Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing*  
553 *systems*, 30, 2017.

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

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

558 Patrick Weber, Jeremy Geiger, and Werner Wagner. Constrained neural network training and its  
559 application to hyperelastic material modeling. *Computational Mechanics*, 68:1179–1204, 2021.

560 Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with  
561 singular value decomposition. In *Interspeech*, pp. 2365–2369, 2013.

562 Baturalp Yalçın, Ziyi Ma, Javad Lavaei, and Somayeh Sojoudi. Semidefinite programming versus  
563 burer–monteiro factorization for matrix sensing. In *Proceedings of the AAAI Conference on*  
564 *Artificial Intelligence*, volume 37, pp. 10702–10710, 2023.

565 Liu Yang, Jifan Zhang, Joseph Shenouda, Dimitris Papailiopoulos, Kangwook Lee, and Robert D  
566 Nowak. PathProx: A proximal gradient algorithm for weight decay regularized deep neural  
567 networks. *arXiv preprint arXiv:2210.03069*, 2022.

568 Yang Yang, Yaxiong Yuan, Avraam Chatzimichailidis, Ruud JG van Sloun, Lei Lei, and Symeon  
569 Chatzinotas. ProxSGD: Training structured neural networks under regularization and constraints.  
570 In *International Conference on Learning Representations*, 2020.

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

574 Jihun Yun, Aurélie C Lozano, and Eunho Yang. Adaptive proximal gradient methods for structured  
575 neural networks. *Advances in Neural Information Processing Systems*, 34:24365–24378, 2021.

576 Alp Yurtsever, Alex Gu, and Suvrit Sra. Three operator splitting with subgradients, stochastic  
577 gradients, and adaptive learning rates. *Advances in Neural Information Processing Systems*, 34:  
578 19743–19756, 2021.

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



## 581 A Additional Details on Matrix Factorization Experiments

### 582 A.1 Matrix Completion under Gaussian Noise

583 We conducted similar experiments to those in Section 2.2 also using noisy measurements: Let  
 584  $b^\natural = \mathcal{A}(X^\natural)$  represent the true measurements, and assume  $b = b^\natural + \omega$ , where  $\omega \in \mathbb{R}^n$  is zero-mean  
 585 Gaussian noise with a standard deviation of  $\sigma = 10^{-2}\|b\|_2$ . The results, shown in Figure SM1,  
 586 remain consistent with the noiseless case.  $UDU^\top$  exhibits implicit bias toward truly low-rank  
 587 solutions, while the classical BM factorization yields approximately low-rank solutions, with the  
 588 spectral decay influenced by initialization.

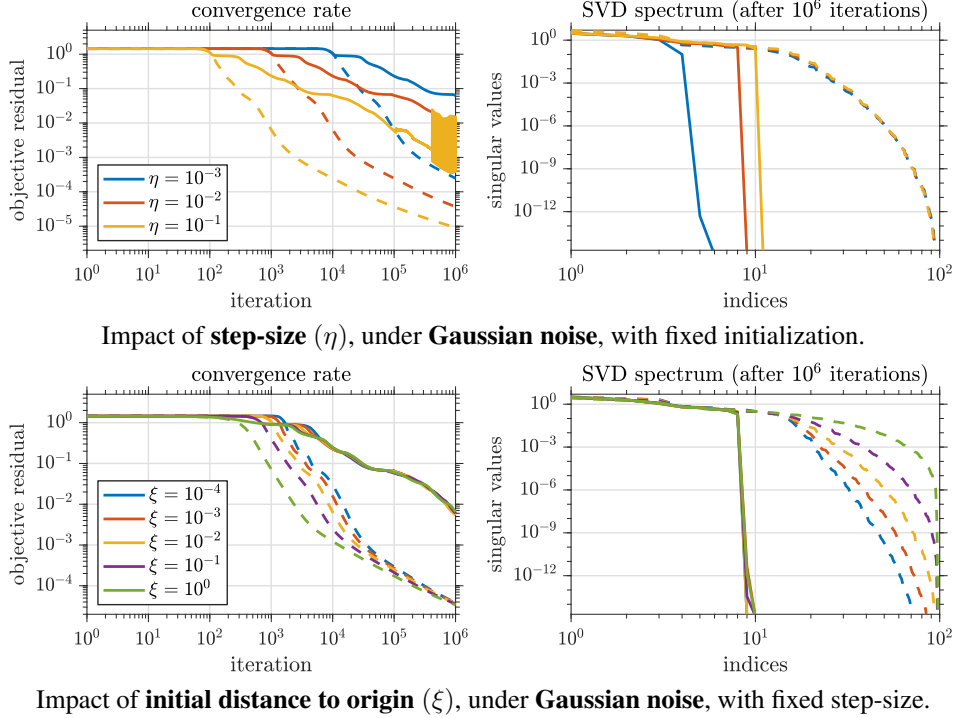


Figure SM1: Impact of step-size and initialization on implicit bias. **Solid lines represent our UDU factorization**, while **dashed lines denote the classical BM factorization**. [Left] Objective residual vs. iterations. [Right] Singular value spectrum after  $10^6$  iterations.

### 589 A.2 Phase Retrieval Image Recovery

590 We conduct a numerical experiment on the matrix sensing problem arising in phase retrieval image  
 591 recovery (Candes et al., 2013). The objective is to recover a signal from its quadratic measurements of  
 592 the form  $y_i = |\langle a_i, x \rangle|^2$ . Although the standard maximum likelihood estimators lead to a non-convex  
 593 optimization problem due to the quadratic terms, the problem can be reformulated as minimizing a  
 594 convex function under a rank constraint in a lifted space. By denoting the lifted variable as  $X = xx^\top$ ,  
 595 the measurements can be expressed as

$$y_i = \langle a_i^\top x, a_i^\top x \rangle = \langle a_i a_i^\top, xx^\top \rangle := \langle A_i, X \rangle,$$

596 which allows us to reformulate the problem as

$$\min_{X \in \mathbb{S}_+^{d \times d}} \frac{1}{2} \|\mathcal{A}(X) - b\|_2^2 \quad \text{subj. to} \quad \text{rank}(X) \leq 1,$$

597 representing a special case of problem (1).

598 In this experiment, we use a gray-scale image of size  $16 \times 16$  pixels, selected from the Pixel Art  
 599 dataset (Elgazar, 2024), as the signal  $x \in \mathbb{R}^n$  to recover, where  $n = 256$ . We generate a synthetic  
 600 measurement system by sampling  $a_1, \dots, a_m$  from the standard Gaussian distribution, with  $m = 2n$ .  
 601 We then solve problems (2) and (4). We initialized  $U_0 \in \mathbb{R}^{n \times n}$  with entries drawn *iid* from the

standard Gaussian distribution, then rescaled to have a unit Frobenius norm. For  $D_0 \in \mathbb{R}^{n \times n}$ , we used the identity matrix. We used step-size  $\eta = \frac{1}{L}$  where  $L$  denotes the smoothness constant. After solving the problem, we recover  $x \in \mathbb{R}^n$  from the lifted variable  $X \in \mathbb{R}^{n \times n}$  by selecting its dominant eigenvector.

Figure SM2 demonstrates that the proposed method consistently identifies low-rank solutions, in line with the results of our other experiments. Figure SM3 displays the recovered image, demonstrating that the proposed method achieves higher quality than BM factorization within the same number of iterations, which can be attributed to the low-rank structure of the solution.

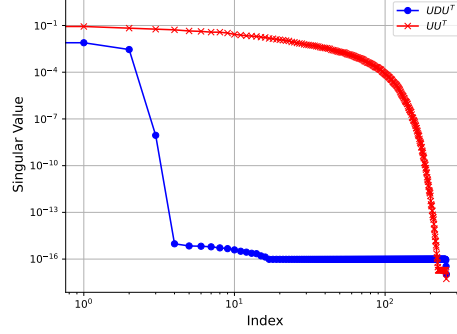


Figure SM2: Comparison of singular value spectrum in recovered image between methods based on  $UDU^T$  and  $UU^T$ .

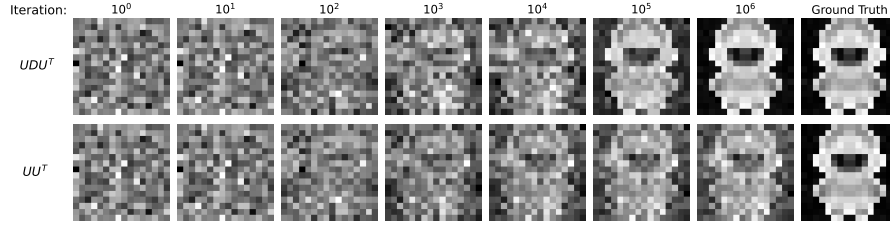


Figure SM3: Comparison of recovered image between methods based on  $UDU^T$  and  $UU^T$ .

### 610 A.3 UDU Factorization Produces Rank-Revealing Solutions

611 The proposed factorization method naturally produces rank-revealing solutions. The columns of  $U$   
 612 converge to zero in certain directions, resulting in truly low-rank solutions. Specifically,  $U$  tends to  
 613 grow along specific directions while the rescaling induced by projection onto the bounded constraint  
 614 shrinks other coordinates. This behavior resembles the mechanism of the power method and provides  
 615 insights into its connection with divergent forces.

616 In Figure SM4, we illustrate the evolution of the column norms of  $U$  from the matrix completion  
 617 experiment described in Section 2.2. Figure SM5 displays the corresponding entries in the diagonal  
 618 factor  $D$ . For comparison, Figure SM6 shows the column norms of  $U$  obtained from the same  
 619 experiment using the standard BM factorization. The results are shown for  $\xi = 10^{-1}$  and  $\eta = 10^{-1}$   
 620 over  $10^5$  iterations, but the findings are similar across other parameter settings. We also repeat  
 621 the experiment with the noisy measurements described in Appendix A.1. The results are shown in  
 622 Figures SM7 to SM9.

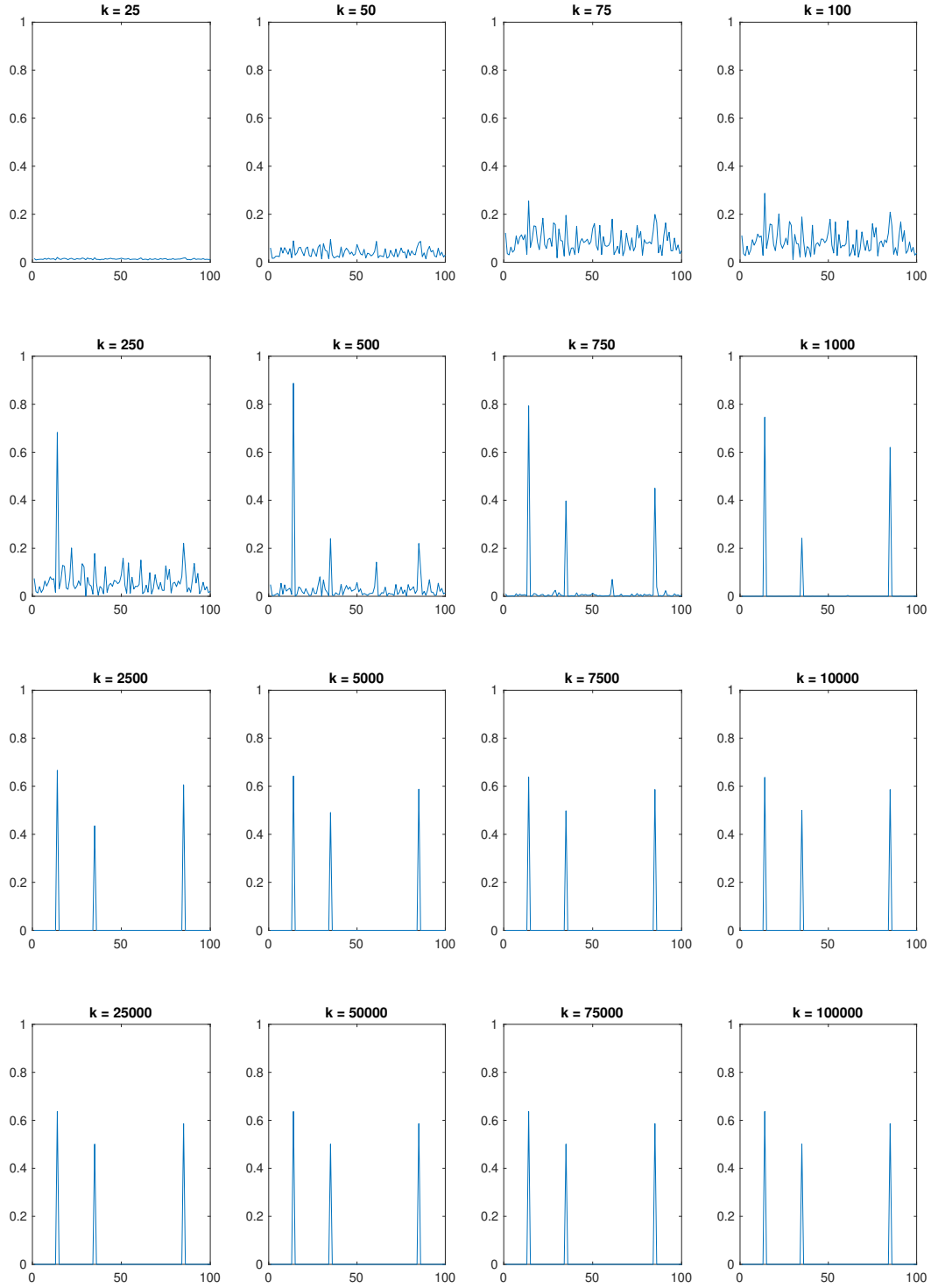


Figure SM4: Evolution of the column norms of  $U$  during the matrix completion experiment using the UDU factorization. The x-axis represents column indices, and the y-axis shows the Euclidean norms of the columns.

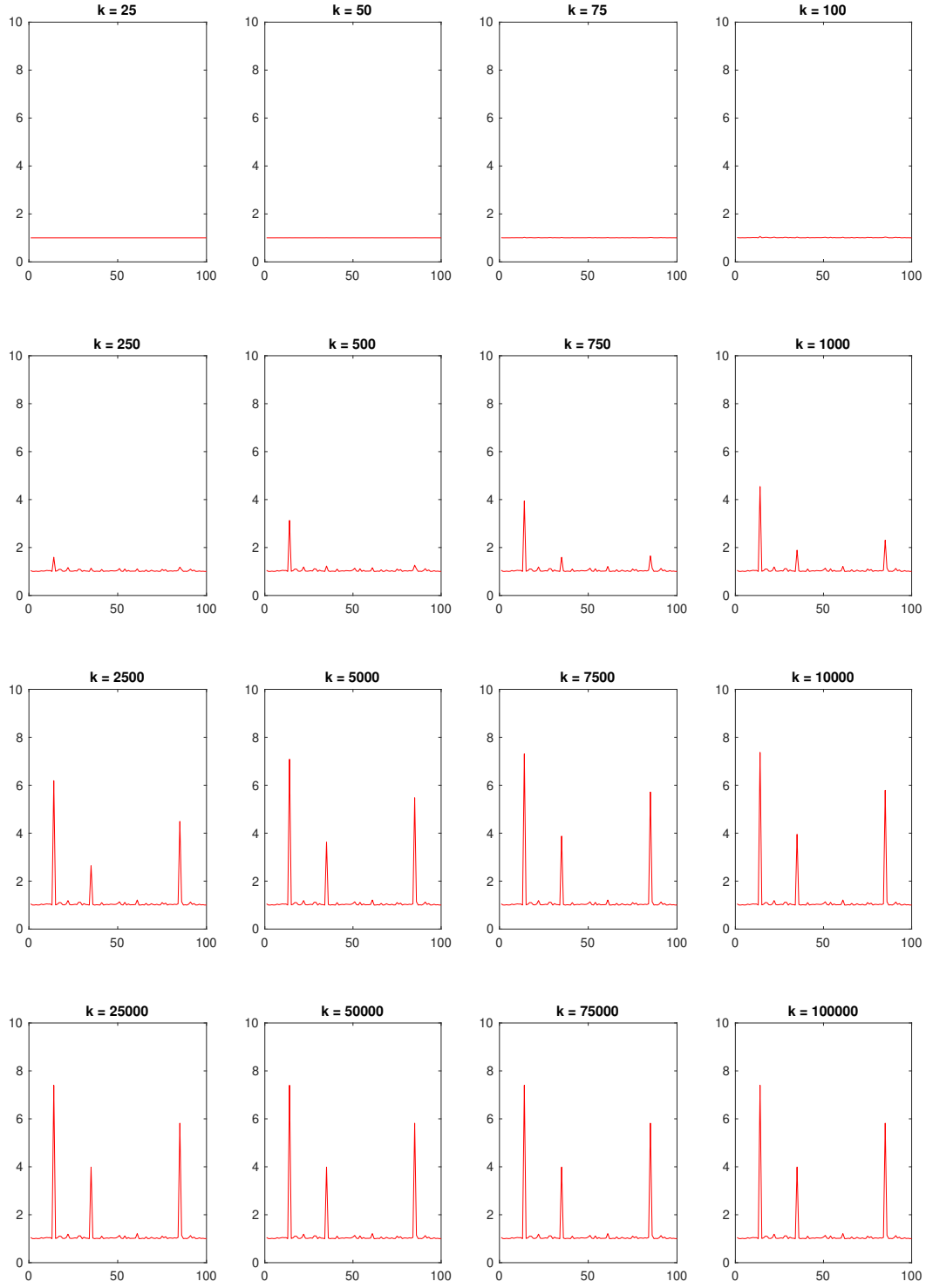


Figure SM5: Evolution of the diagonal entries of  $D$  during the matrix completion experiment using the UDU factorization. The x-axis represents indices.

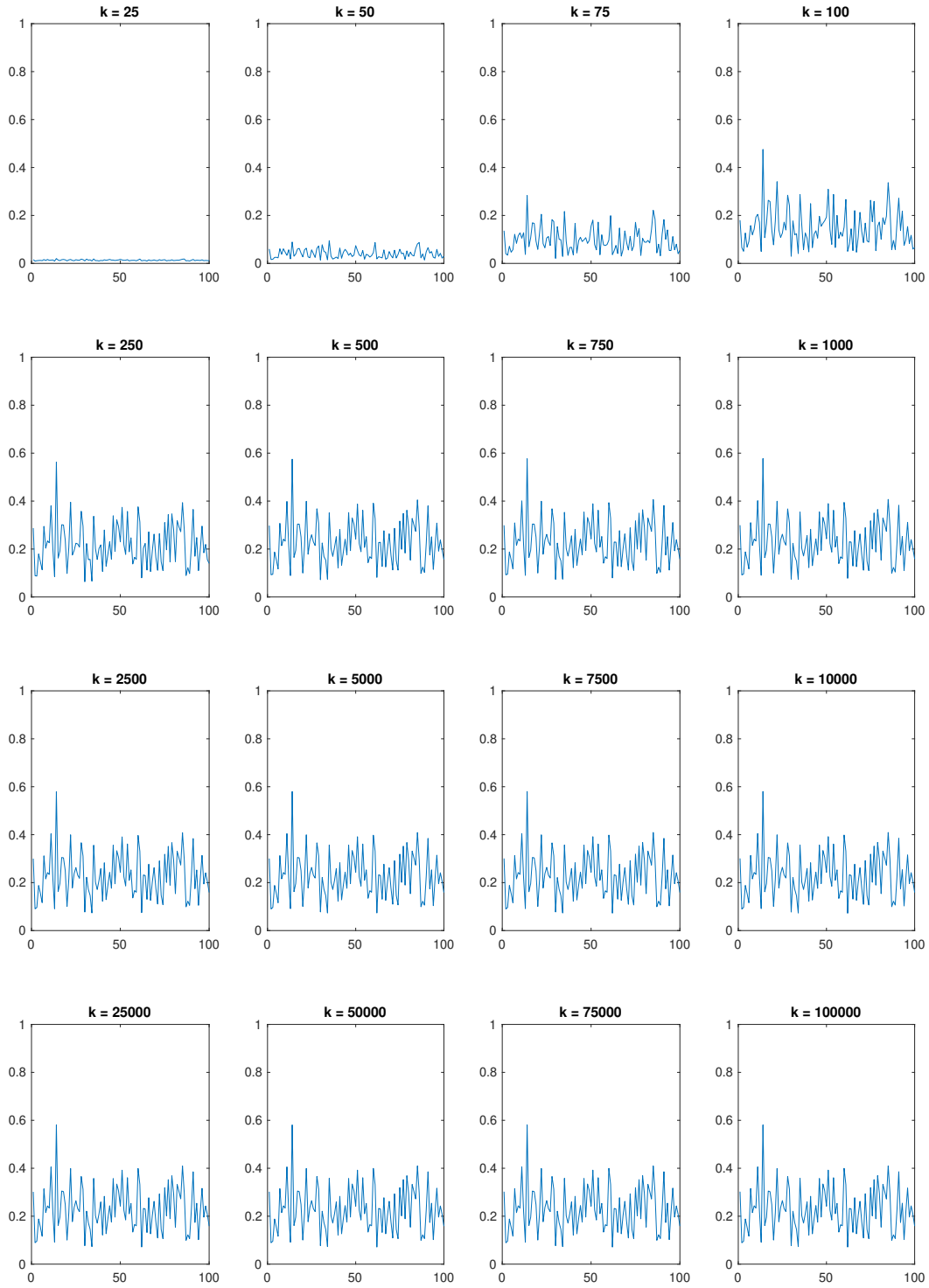


Figure SM6: Evolution of the column norms of  $U$  during the matrix completion experiment using the standard BM factorization, shown for comparison. The x-axis represents column indices, and the y-axis shows the Euclidean norms of the columns.

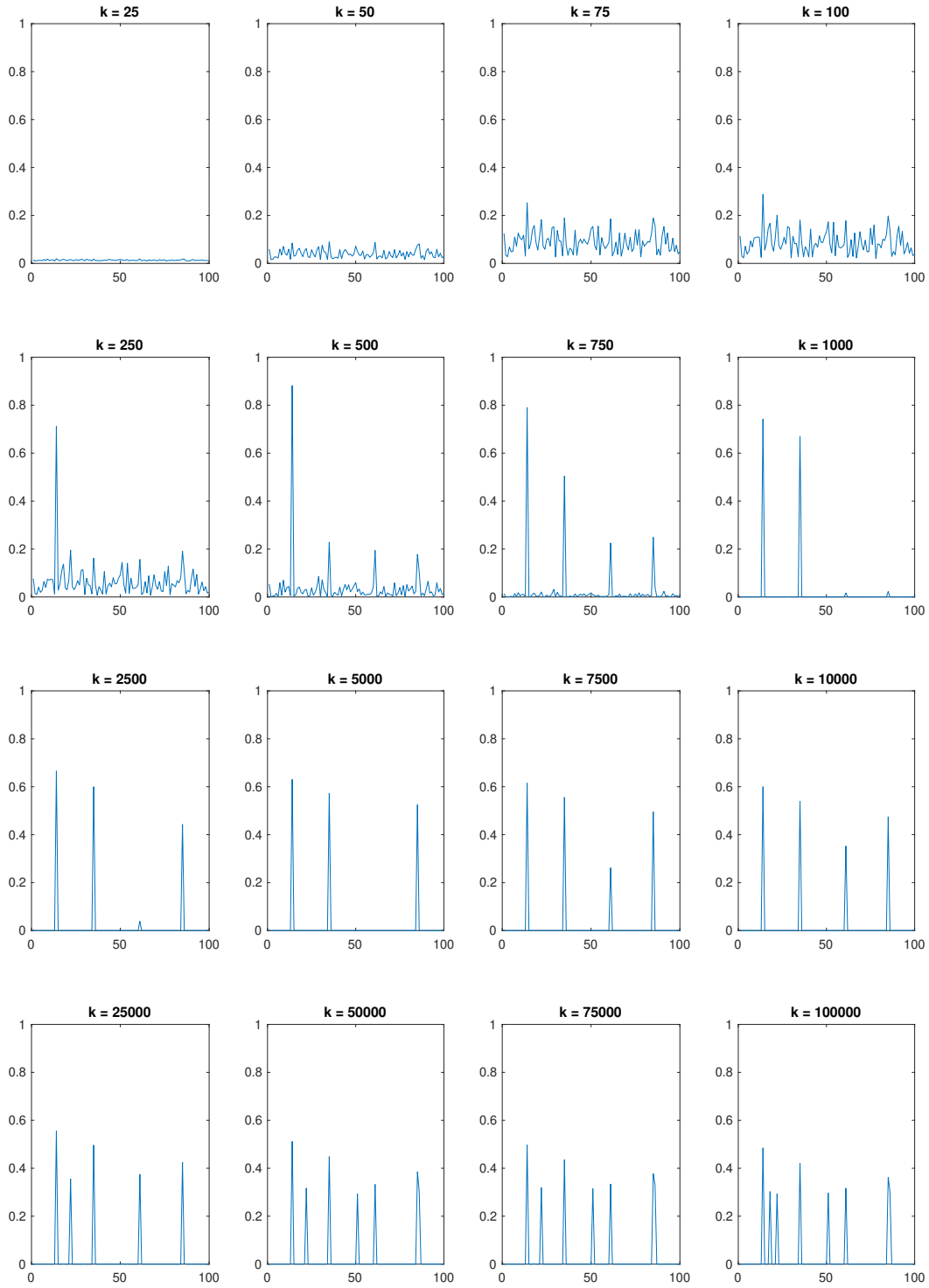


Figure SM7: Evolution of the column norms of  $U$  during the matrix completion experiment using the UDU factorization with **noisy** measurements. The x-axis represents column indices, and the y-axis shows the Euclidean norms of the columns.



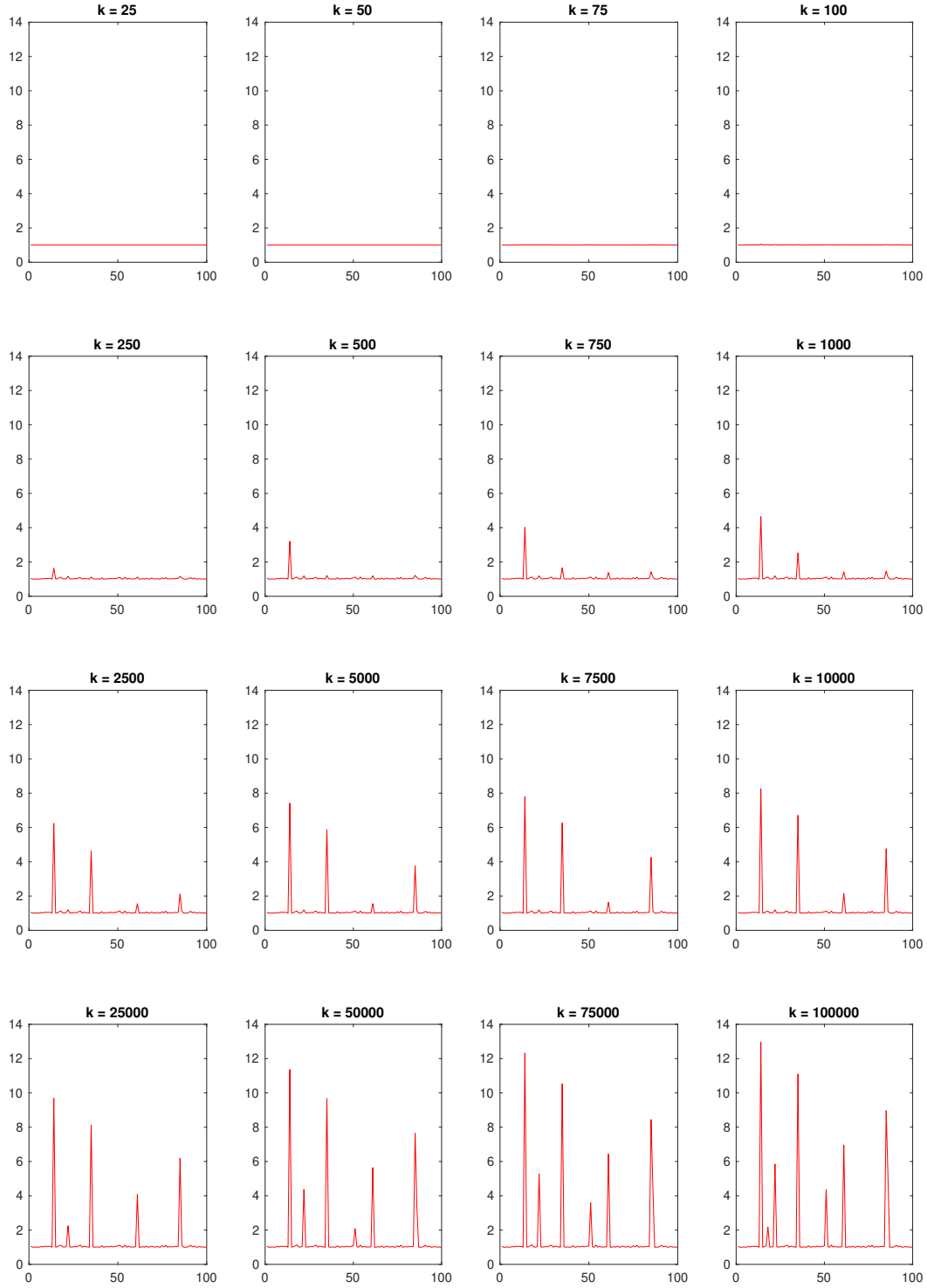


Figure SM8: Evolution of the diagonal entries of  $D$  during the matrix completion experiment using the UDU factorization with **noisy** measurements. The x-axis represents indices.

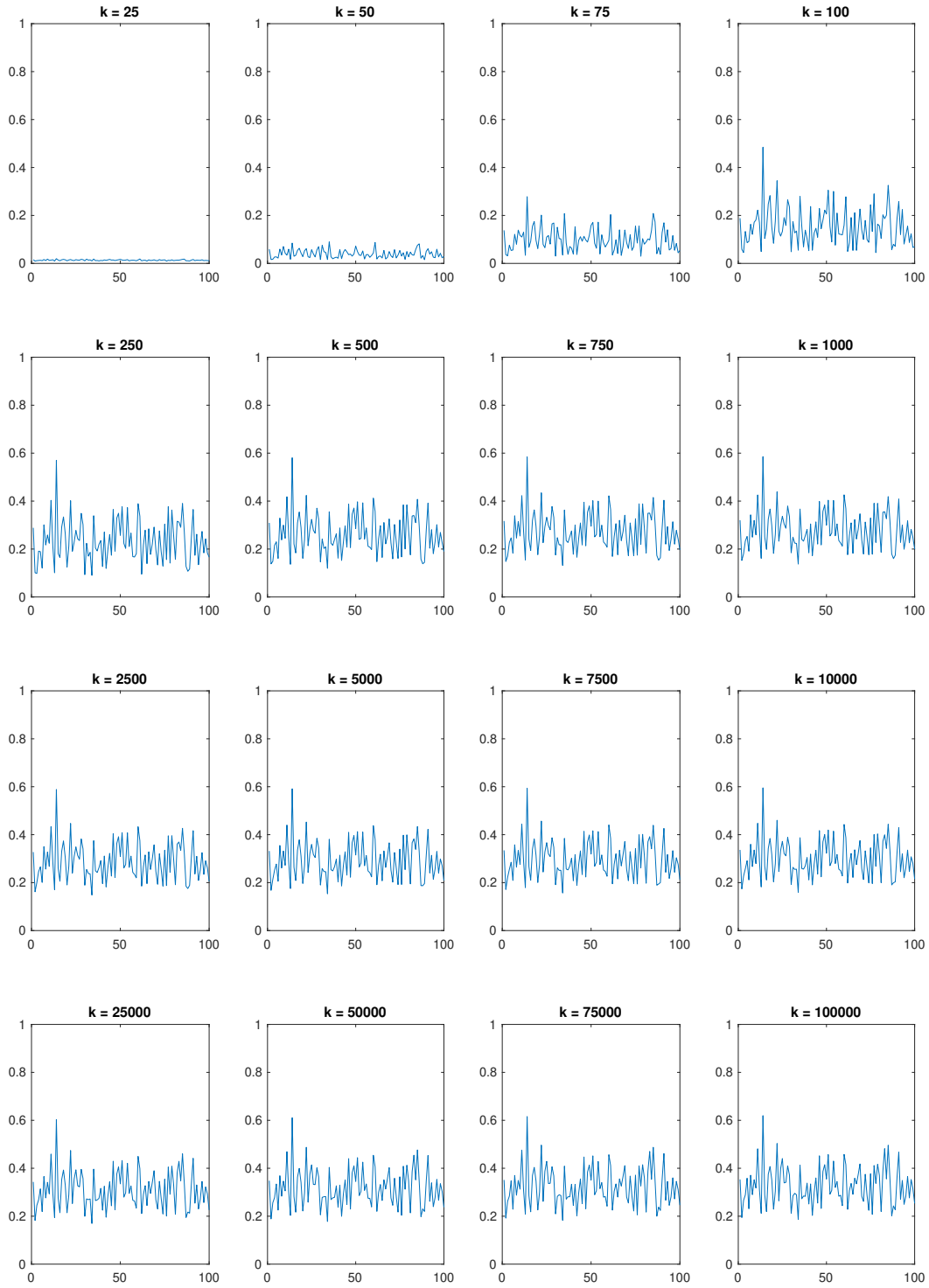


Figure SM9: Evolution of the column norms of  $U$  during the matrix completion experiment using the standard BM factorization with **noisy** measurements, shown for comparison. The x-axis represents column indices, and the y-axis shows the Euclidean norms of the columns.

#### 623 A.4 Additional Details on the Fixed Point Analysis

624 In [Section 2.3](#), we provided insights into the inner workings of the UDU framework on matrix  
625 factorization problems based on a fixed-point analysis. Recall the algorithm steps:

$$\begin{aligned} U_{k+1} &= \Pi_U(\bar{U}_{k+1}) \quad \text{where} \quad \bar{U}_{k+1} = U_k - 2\eta \nabla f(U_k D_k U_k^\top) U_k D_k \\ D_{k+1} &= \Pi_D(\bar{D}_{k+1}) \quad \text{where} \quad \bar{D}_{k+1} = D_k - \eta U_k^\top \nabla f(U_k D_k U_k^\top) U_k \end{aligned}$$

626 Let  $(U, D)$  denote a fixed point of this algorithm and let  $X = UDU^\top$ . Suppose  $\|\bar{U}\| \leq \alpha$ . In this  
627 case, the projection step  $\Pi_U$  acts as an identity:

$$U = \bar{U} = U - 2\eta \nabla f(X)UD \implies \nabla f(X)UD = 0 \quad (10)$$

628 Otherwise, if we assume  $\|\bar{U}\| \geq \alpha$ , then the projection  $\Pi_U$  has a normalization effect:

$$U = \frac{\alpha}{\|\bar{U}\|} \bar{U} = \frac{\alpha}{\|\bar{U}\|} (U - 2\eta \nabla f(X)UD)$$

629 which we can reorganize as

$$\nabla f(X)UD = -\beta U \quad \text{where} \quad \beta := \frac{\|\bar{U}\| - \alpha}{2\alpha\eta}. \quad (11)$$

630 Denoting the columns of  $U$  by  $u_j$  and the diagonal entries of  $D$  by  $\lambda_j$ , we express the conditions (10)  
631 and (11) in terms of the individual components as presented in conditions (a) and (b) in [Section 2.3](#).

632 Next, we investigate the  $D$  component. Note that the projection  $\Pi_D$  operates in an entry-wise  
633 separable manner; it maps all non-diagonal entries to 0, and takes the positive part for the diagonal  
634 entries. It maps all off-diagonal entries to zero and applies the positive-part operator to the diagonal  
635 entries. Hence, it suffices to examine only the diagonal elements.

636 Suppose  $\bar{\lambda}_j \leq 0$ . Then  $\lambda_j = \max(\bar{\lambda}_j, 0) = 0$ , and

$$\bar{\lambda}_j = \lambda_j - \eta u_j^\top \nabla f(X)u_j = -\eta u_j^\top \nabla f(X)u_j \implies u_j^\top \nabla f(X)u_j \geq 0. \quad (12)$$

637 Otherwise, if  $\bar{\lambda}_j > 0$ , then  $\lambda_j = \max(\bar{\lambda}_j, 0) = \bar{\lambda}_j$ , hence

$$\lambda_j = \bar{\lambda}_j = \lambda_j - \eta u_j^\top \nabla f(X)u_j \implies u_j^\top \nabla f(X)u_j = 0. \quad (13)$$

638 The conditions in (12) and (13) correspond to cases (c) and (d) in [Section 2.3](#).

## B Additional Details on Neural Network Experiments

### B.1 Design Variants of the UDV Architecture

When designing our network architecture, we considered four variants, including the one presented in Section 3. The three additional models were defined by the following sets of constraints:

$$\sum_{j=1}^m \|\mathbf{u}_j\|_2^2 \leq 1, \quad \sum_{j=1}^m \|\mathbf{v}_j\|_2^2 \leq 1 \quad (\text{UDV-s})$$

$$\|\mathbf{u}_j\|_2^2 \leq 1, \quad \|\mathbf{v}_j\|_2^2 \leq 1 \quad w_j \geq 0; \quad \text{for all } j = 1, \dots, m \quad (\text{UDV-v1})$$

$$\|\mathbf{u}_j\|_2^2 \leq 1, \quad \|\mathbf{v}_j\|_2^2 \leq 1; \quad \text{for all } j = 1, \dots, m \quad (\text{UDV-v2})$$

In detail, UDV-s is identical to UDV but omits the non-negativity constraints on the diagonal layer. UDV-v1, on the other hand, enforces row/column-wise norm constraints instead of the Frobenius norm used in UDV, while retaining the non-negativity constraints on the diagonal elements. Finally, UDV-v2 is identical to UDV-v1 but without the non-negativity constraints on the diagonal layer.

We only report results from UDV, as defined in equation (7), in the main text. However, we provide results for the other variants in the Appendices for completeness and to illustrate the impact of different constraint settings on model performance.

Figure SM11 and Figure SM12 show that UDV consistently finds a low-rank solution. Generally, UDV exhibits the most pronounced decaying pattern in singular values. Additionally, we extended our experiments to include full-batch training on the HPART and NYCTTD datasets. Although full-batch training converges more slowly than stochastic (mini-batch) methods, it exhibits a similar singular value decay pattern, confirming our earlier observations.

### B.2 Comparison of Model Performance with the MBGDM Algorithm

In Section 3.1, we presented the performance of each model (UDV, UV, and UV-ReLU) in terms of generalization power (measured by validation loss or validation accuracy) along with the corresponding singular value spectrum, as shown in Table 1 and Figure 3. The results were obtained by selecting the optimal algorithm and learning rate pair for each model. Here, we conduct the same experiment, but focus exclusively on the MBGDM algorithm. The results, presented in Table SM1 and Figure SM10 are similar to the ones in Section 3.1, showing similar trends, highlighting consistency across methods.

Table SM1: Model performance using MBGDM. 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	$1.345 \times 10^{-3}$	$5.248 \times 10^{-6}$	M: 99.67%	E: 99.63%	R: 99.74%
	MBGDM: $10^{-1}$	MBGDM: $10^{-1}$	MBGDM: $10^{-2}$	MBGDM: $10^{-1}$	MBGDM: $10^{-2}$
UV	$1.337 \times 10^{-3}$	$5.259 \times 10^{-6}$	M: 99.69%	E: 99.59%	R: 99.56%
	MBGDM: $10^{-2}$	MBGDM: $10^{-1}$	MBGDM: $10^{-2}$	MBGDM: $10^{-1}$	MBGDM: $10^{-1}$
UV-ReLU	$1.244 \times 10^{-3}$	$5.264 \times 10^{-6}$	M: 99.63%	E: 99.68%	R: 99.66%
	MBGDM: $10^{-1}$	MBGDM: $10^{-1}$	MBGDM: $10^{-2}$	MBGDM: $10^{-1}$	MBGDM: $10^{-1}$

### B.3 Additional Details on the Pruning Experiment

The SVD-based pruning experiments yield conclusions consistent with the singular value decay pattern. UDV typically exhibits the fastest decay, leading to more compact models while maintaining performance, as shown in Figure SM13.

We observed that the learning rate can have some impact on the singular value decay pattern. In particular, a very large learning rate may cause oscillations in both training and validation loss, yet may

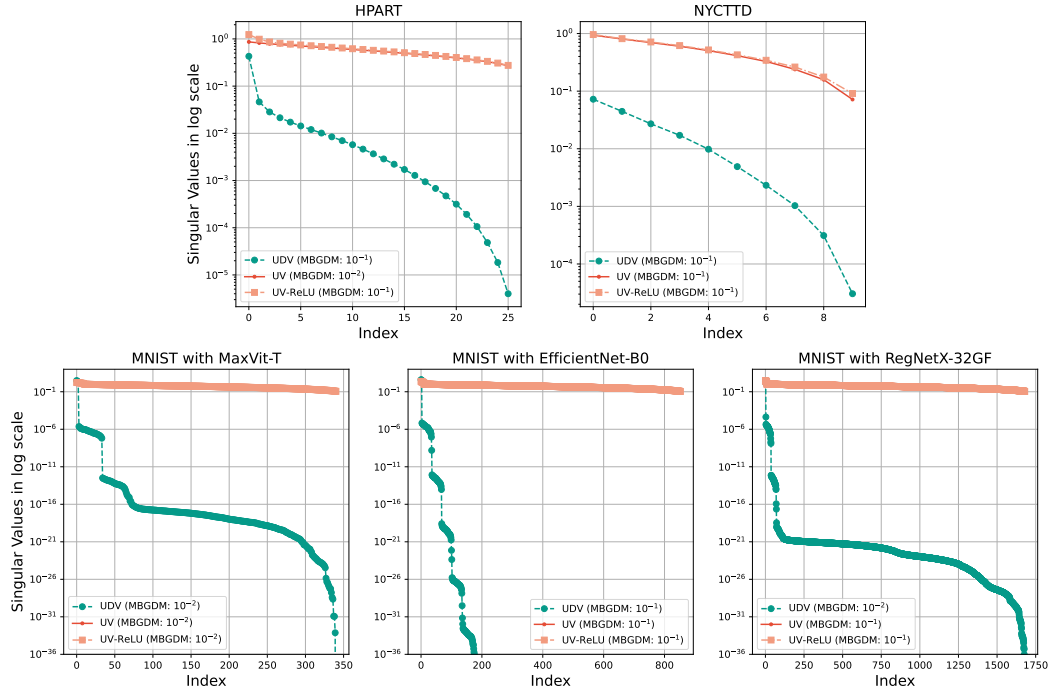


Figure SM10: Singular value spectrum corresponding to Table SM1.

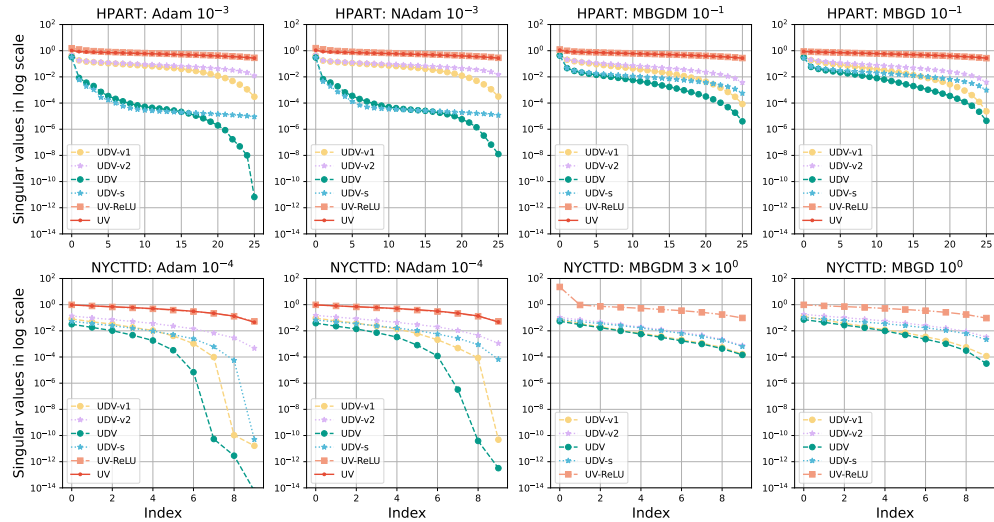


Figure SM11: Comparison of singular value pattern among all UDV variants, UV-ReLU and UV on the regression tasks.

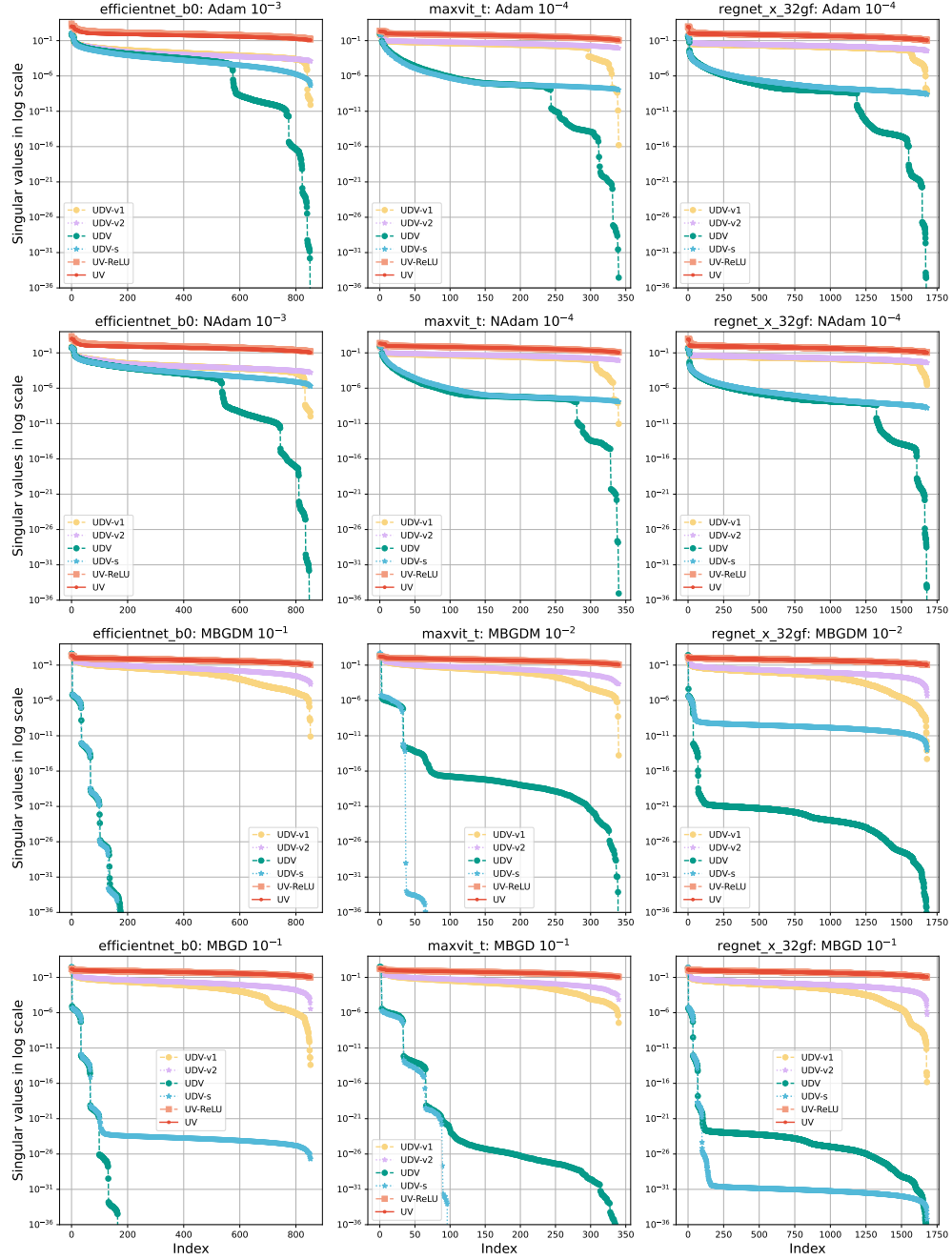


Figure SM12: Comparison of singular value pattern among all UDV variants, UV-ReLU and UV on the MNIST dataset.



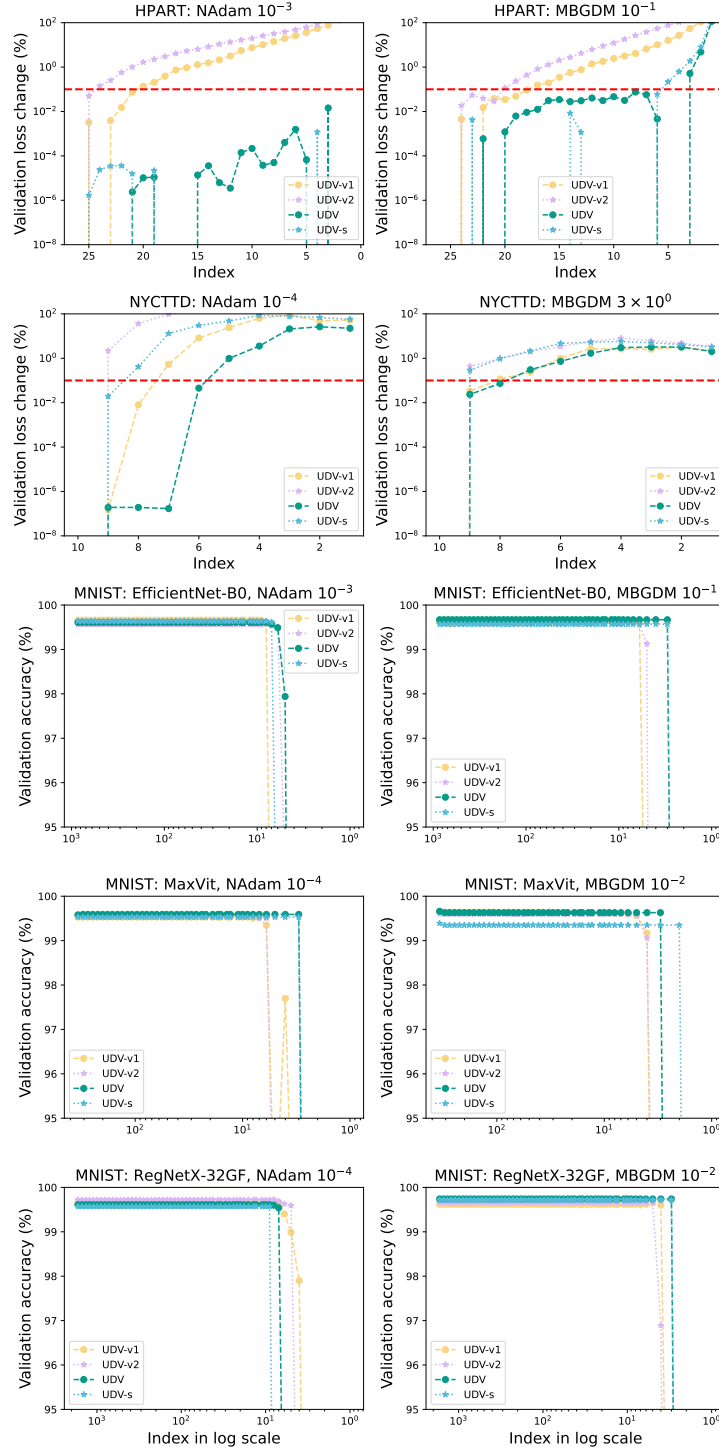


Figure SM13: The performance of SVD-based pruning. The index in x-axis represents the number of neurons in the diagonal layer after pruning. For the HPART and NYCTTD datasets, the validation loss change indicates how much worse the pruned model performs compared to the baseline (the model before pruning), expressed as a percentage ( $\frac{\text{loss}_{\text{pruned}} - \text{loss}_{\text{baseline}}}{\text{loss}_{\text{baseline}}} \times 100\%$ ). Note that the pruned model may outperform the baseline, but negative values cannot be displayed on a logarithmic scale. The red dashed line denotes the 0.1% threshold, indicating negligible performance sacrifice. For the MNIST dataset, the results show the validation accuracy after pruning the model.

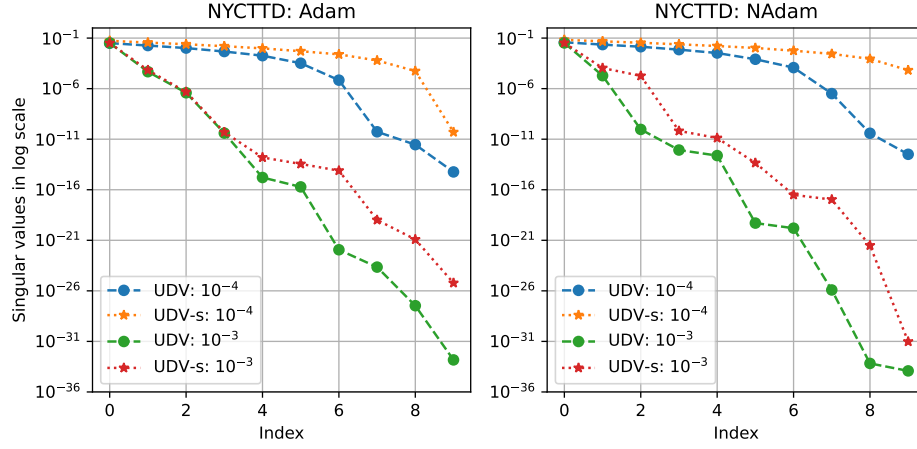


Figure SM14: Differences in singular value patterns across varying learning rates.

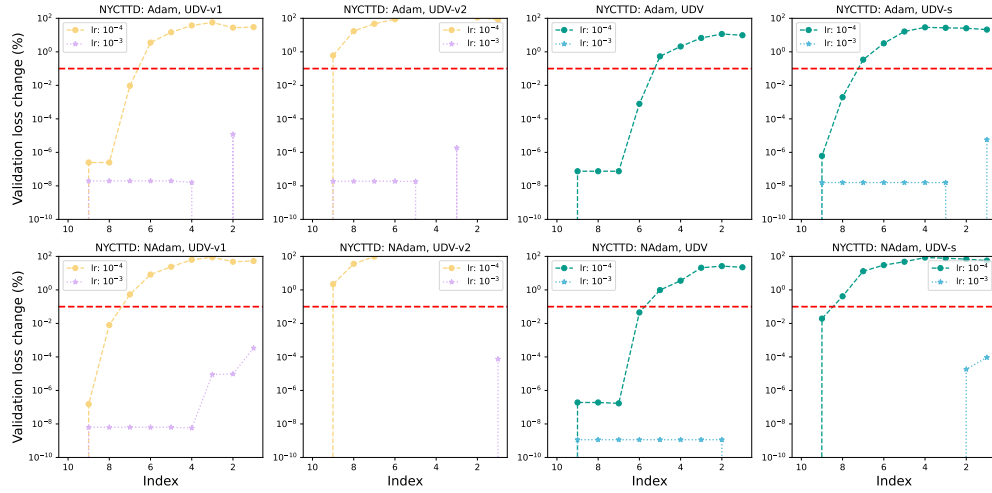


Figure SM15: Different learning rates lead different pruning performance. the validation loss change indicates how much worse the pruned model performs compared to the baseline (the model before pruning), expressed as a percentage ( $\frac{loss_{pruned} - loss_{baseline}}{loss_{baseline}} \times 100\%$ ). Note that the pruned model may outperform the baseline, but negative values cannot be displayed on a logarithmic scale. The red dashed line denotes the 0.1% threshold, indicating negligible performance sacrifice.

669 result in a rapid decay of the spectrum. Conversely, a small learning rate may lead to a less pronounced  
 670 spectral decay but still can yield a comparable validation loss to that of the optimal learning rates. For  
 671 example, the difference in validation losses between the Adam optimizer with learning rates of 1e-3  
 672 and 1e-4 was negligible, yet the singular value spectra differed (see Figure SM14). This discrepancy  
 673 also affects the performance of the pruning experiments (see Figure SM15).

674 When the learning rate is close to the optimal value, a ‘stair-step’ pattern (see Figure SM16) can be  
 675 observed on the loss or accuracy curve in the classification task. It could be attributed to the model  
 676 continuously searching for a low-rank solution, increasing the rank gradually.

#### 677 B.4 Incorporating ReLU into UDV

678 To explore whether non-linear activation functions, particularly ReLU, can be used in our UDV  
 679 framework, we conducted an additional experiment.

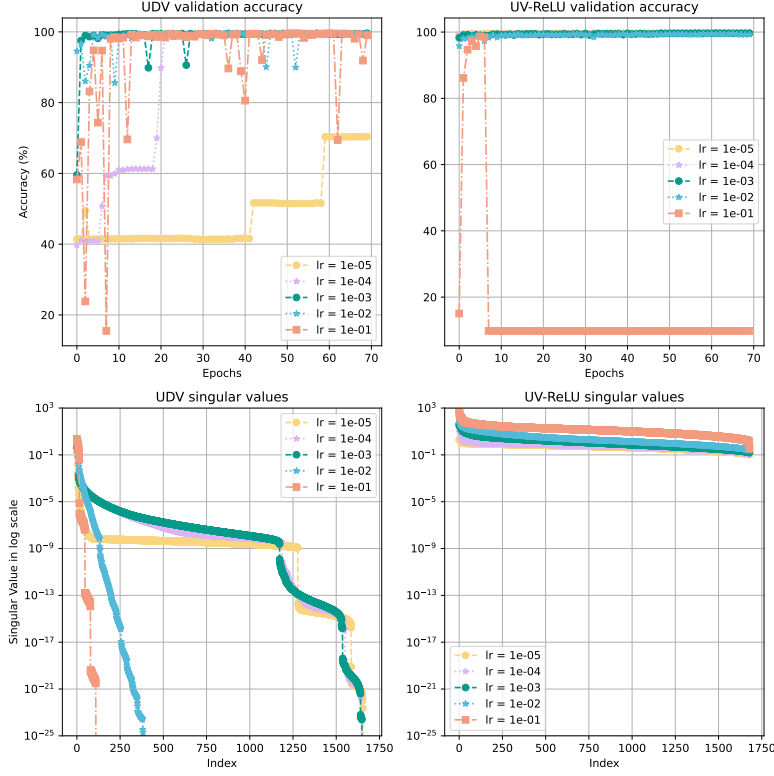


Figure SM16: The ‘stair-step’ accuracy curve in classification (RegNetX-32GF with Adam). The sub-figures in the second row correspond to the respective singular value spectra, indicating that UDV consistently seeks low-rank solutions regardless of the learning rate.

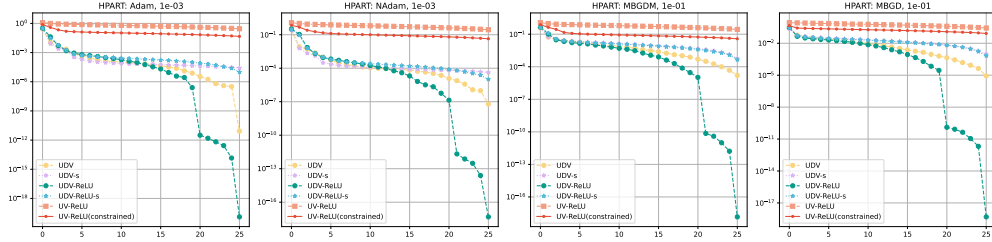


Figure SM17: ReLU in UDV: Singular value pattern on HPART dataset.

Building on the standard ReLU, we first introduced norm constraints,  $\sum_{j=1}^m \|\mathbf{u}_j\|_2^2 \leq 1$  and  $\sum_{j=1}^m \|\mathbf{v}_j\|_2^2 \leq 1$ , to the layers both before and after ReLU, denoted as *ReLU(constrained)*. Next, we integrated ReLU into the diagonal layer of UDV. When the non-negativity constraints are applied on the diagonal layer, we refer to the model as UDV-ReLU; otherwise, it is called UDV-ReLU-s.

We conducted a preliminary experiment to verify the feasibility of incorporating ReLU into the UDV, leaving a comprehensive study to future work. We derived the optimizers and learning rates from [Tables SM2 to SM5](#), but only used the HPART dataset for regression and the MNIST for classification.

[Figure SM17](#) and [Figure SM18](#) showed that UDV-ReLU and UDV-ReLU-s exhibit a similar (or even more pronounced) decaying pattern in singular values as observed in the proposed UDV. The ‘stair-step’ accuracy curve was also observed in these models.

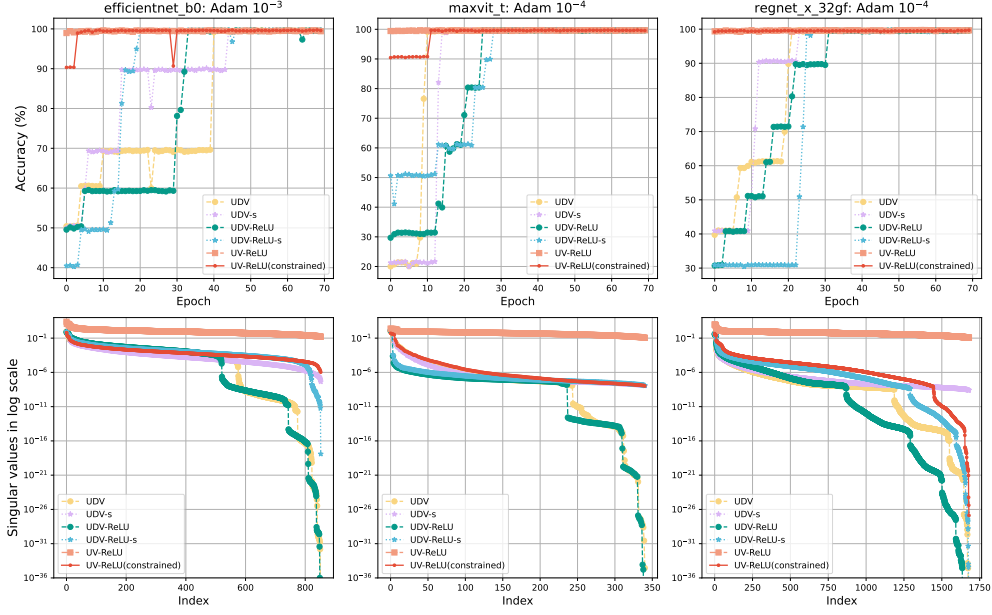


Figure SM18: ReLU in UDV: Validation accuracy (the first row) and the singular value pattern (the second row) on MNIST dataset. Faster convergence can be achieved by fine-tuning the learning rate.

## 691 B.5 Comparison of UDV and Three-Layer Fully Connected Networks

692 We extend the experiments to compare the UDV framework with three-layer fully connected neural  
 693 networks. To highlight the critical role of the UDV structure and its constraints, we include the  
 694 following models in the comparison: the proposed UDV model, a model based on the UDV structure  
 695 but without the constraints (UDV unconstrained), a fully connected three-layer neural network (UFV),  
 696 and the standard UV model as the baseline.

697 The results, also included in Tables SM2 to SM5, are shown in Figure SM19 and Figure SM20.  
 698 These findings demonstrate that the singular value decay in UDV unconstrained, UFV, and UV is  
 699 significantly slower than in the proposed UDV framework. This underscores the critical role of the  
 700 diagonal layer and constraints in facilitating low-rank solution identification within the proposed  
 701 structure.

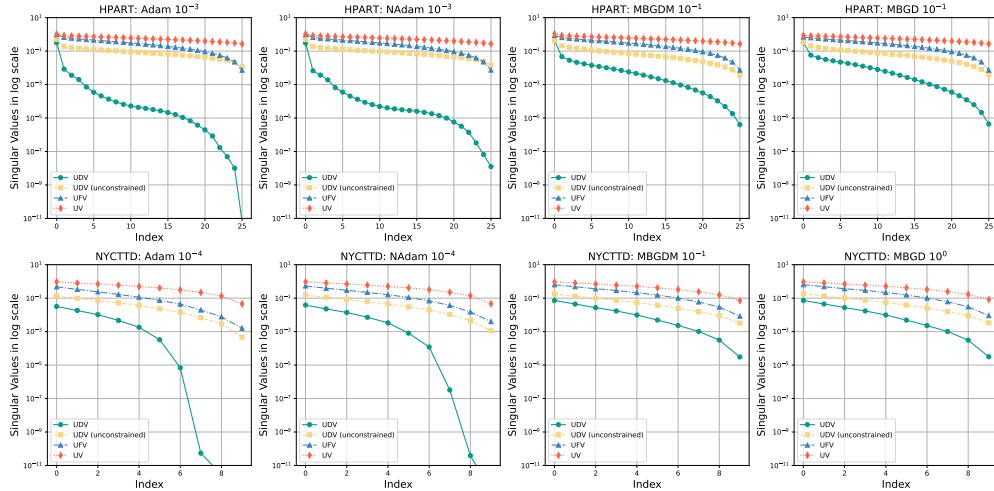


Figure SM19: Comparison of singular value spectrum among UDV, UDV (unconstrained), UFV and UV on the regression tasks.

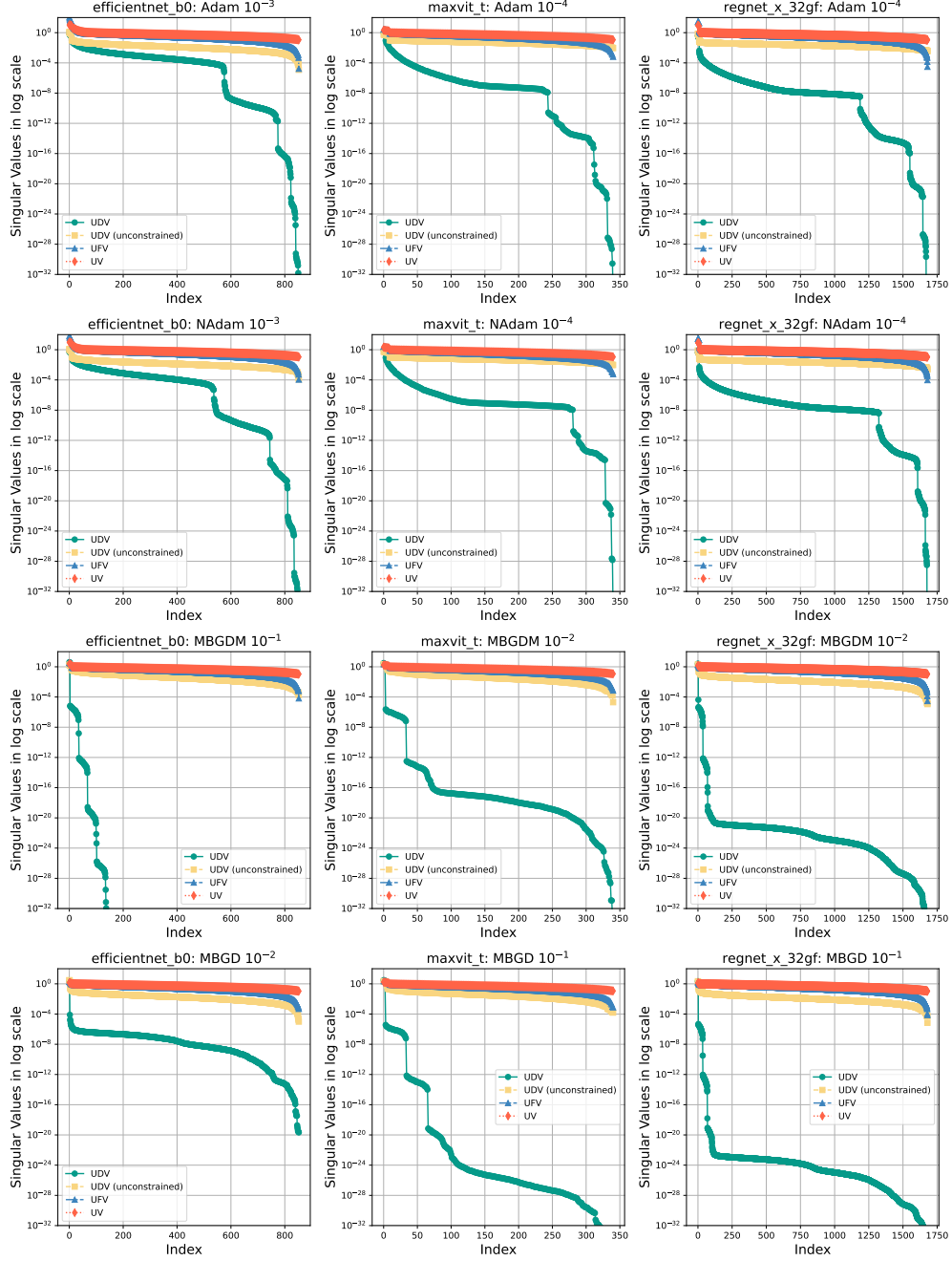


Figure SM20: Comparison of singular value spectrum among UD, UD (unconstrained), UF and UV on the classification datasets (MNIST).

## 702 B.6 Comparing the Effects of UDV and Weight Decay Regularization

703 We conducted a preliminary experiment to compare the singular value decay in two-layer (UV)  
 704 and three-layer (UFV) fully connected network blocks with weight decay regularization to that  
 705 observed in the UDV framework. We focused on the regression task with the HPART dataset. For the  
 706 regularization parameter ( $\gamma$ ), we tested values  $10^{-6}, 10^{-5}, \dots, 10^{-1}$ .

707 Increasing  $\gamma$  led to a faster decay in the singular value spectrum but at the cost of higher training  
 708 and validation losses. Figure SM21 illustrates an example where we selected the largest  $\gamma$  values for  
 709 which the validation loss remained within 10% of the UDV baseline. The resulting singular value  
 710 decay in the UV and UFV models was less pronounced than in the UDV model. Further increasing  $\gamma$   
 711 to match or exceed the spectral decay of the UDV model resulted in significantly worse validation  
 712 loss compared to the UDV network. The complete results are presented in Figures SM22 and SM23.

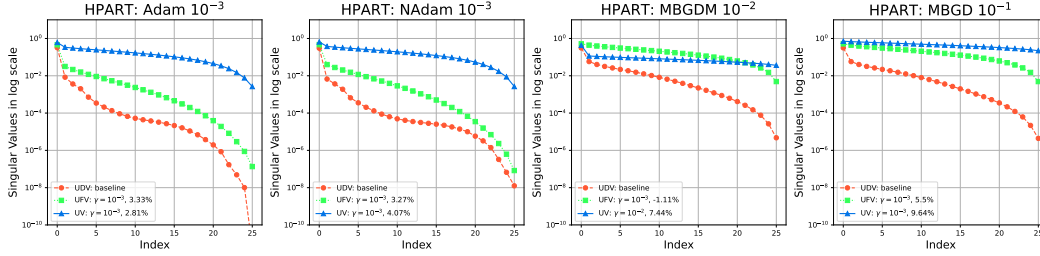


Figure SM21: Comparison of singular value spectrum among the proposed UDV, UFV and UV (with weight decay) on the HPART dataset. In addition to the UDV baseline, the UFV and UV models include the regularization parameter  $\gamma$  and the relative change in validation performance. The relative change is defined as  $(\frac{\text{loss}_{\text{model}} - \text{loss}_{\text{baseline}}}{\text{loss}_{\text{baseline}}} \times 100\%)$ , representing how much worse the model performs compared to the UDV baseline.

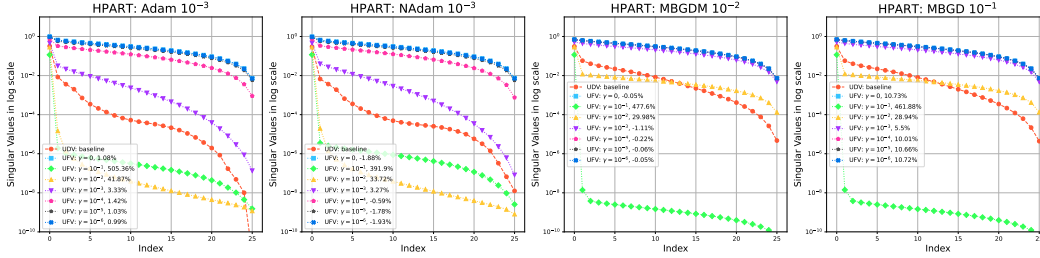


Figure SM22: Extension of Figure SM21. Comparison of singular value spectrum between the proposed UDV and UFV (with weight decay) on the HPART dataset. Regularization parameter  $\gamma = 0$  indicates that no weight decay is applied.

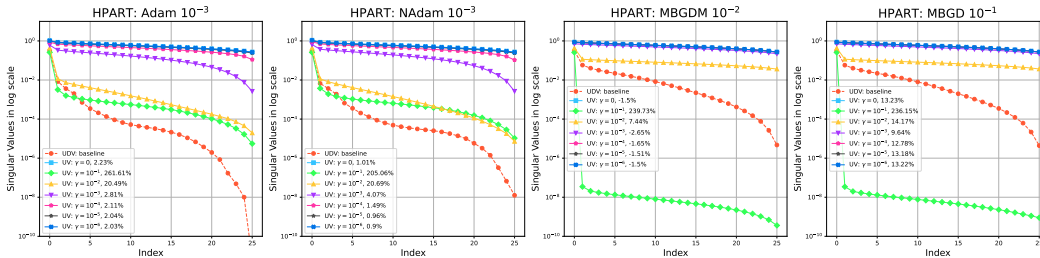


Figure SM23: Extension of Figure SM21. Comparison of singular value spectrum between the proposed UDV and UV (with weight decay) on the HPART dataset. Regularization parameter  $\gamma = 0$  indicates that no weight decay is applied.



## B.7 A Toy Example of Causal Language Modeling Using a Pre-trained LLaMA-2 Model with LoRA-Based UDV Fine-Tuning

To further broaden the application scope of the proposed method, we integrate UDV into the pre-trained model using the LoRA framework and evaluate its effectiveness on the causal language modeling task (i.e., a generative task where a model predicts the next token in a sequence based solely on past tokens). Experiments are conducted using the LLaMA-2-7B model (Touvron et al., 2023) and the WikiText-2 dataset (Merity et al., 2016), which is a common benchmark test.

LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning technique that inserts trainable low-rank matrices into pre-trained models (Hu et al., 2022). By freezing the original model weights and only training a small number of additional parameters, LoRA enables efficient adaptation to downstream tasks while significantly reducing computational and storage costs. Specifically, LoRA introduces trainable low-rank matrices  $B \in \mathbb{R}^{d \times r}$  and  $A \in \mathbb{R}^{r \times k}$  as a side path to the frozen pre-trained weight matrix  $W_0$ , and expresses the forward pass as  $h = W_0x + BAx$ , where  $x$  is the input. Here we consider  $d = k$  for simplicity.

By inserting a diagonal matrix  $D$  between the matrices  $B$  and  $A$ , the LoRA structure naturally extends to the LoRA-based UDV structure, as illustrated in Figure SM24. After the weight update of  $U$ ,  $D$  and  $V$ , constraints described in Equation (7) are applied to the LoRA-based UDV structure. In practice, the diagonal matrix can be efficiently implemented using a vector  $d \in \mathbb{R}^{1 \times r}$ , further reducing computational and memory overhead. Consequently, the LoRA-based UDV introduces a negligible increase in the number of trainable parameters compared to the original LoRA structure, as it adds only  $r$  parameters per LoRA layer replacement.

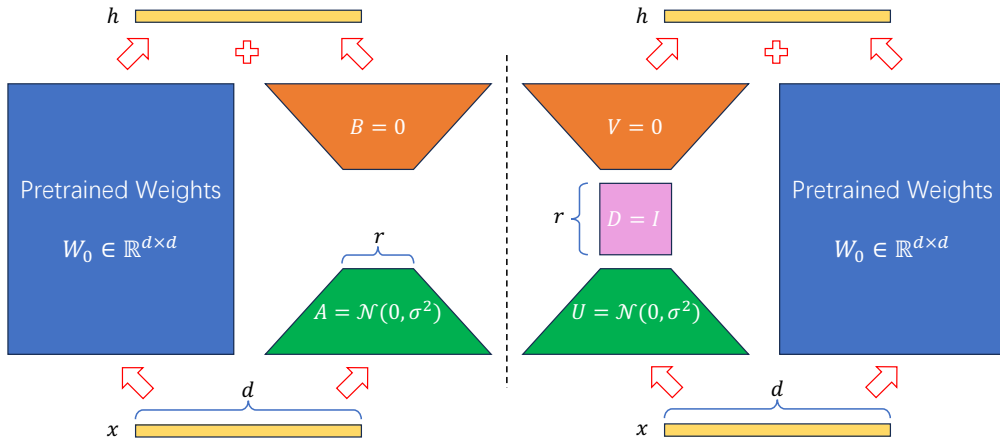


Figure SM24: LoRA forward pass (Left) and LoRA-based UDV forward pass (Right). The inputs and pre-trained weights are identical in both cases. Matrices with the same fill color indicate matching shapes.

We initialized both the LoRA and LoRA-based UDV structures from the same starting point:  $B = V = 0$ ,  $A = U = \mathcal{N}(0, 0.01)$  and  $D = I$ , where  $I$  is the identity matrix. Mini-batch gradient descent (MBGD) with learning rates  $lr = 0.001$ , using an effective batch size of 32, was applied to better observe the effects of the imposed constraints. Following the standard LoRA setup, only the linear layers associated with the Query ( $Q$ ) and Value ( $V$ ) projections in the attention mechanism were replaced with LoRA and LoRA-based UDV structures. We evaluated the models using rank values  $r \in \{4, 8, 16, 32, 64\}$ , where the dimensionality of  $d$  depends on the shape of the original layer. Both structures were fine-tuned for 30 epochs, and then pruning was applied as described in Section 3.1.3.

Instead of applying uniform pruning across all LoRA-based UDV structures, we set several thresholds on the retained energy—defined as the sum of the squared singular values—to dynamically prune each UDV block based on its individual importance. Note that the pruning can be applied to the LoRA structure in the same manner as the LoRA-based UDV structure by temporarily inserting an frozen identity matrix as a diagonal layer.

Perplexity (i.e., the exponential of the average negative log-likelihood) is used as the primary evaluation metric for this experiment, and lower perplexity indicates better language modeling

performance. Furthermore, the pruning performance was demonstrated by the perplexity change in terms of the compression ratio (i.e., the ratio between the number of trainable parameters of the compressed structure and the original structure)

Since the hard constraints slower the convergence, and its fine-tuning performance is slightly lower than that of the original LoRA structure within the same number of epochs, as shown in Figure SM25 shown. However, the pruning results (see Figure SM26) demonstrate that the LoRA-based UDV is significantly more robust to high compression ratios. Even at a compression ratio of 0.1, the LoRA-based UDV is able to maintain performance effectively.

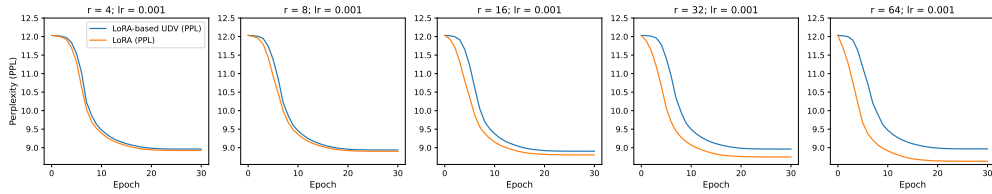


Figure SM25: Comparison of perplexity between LoRA and LoRA-based UDV during training. The perplexity at epoch 0 reflects the performance of the pre-trained model before any fine-tuning.

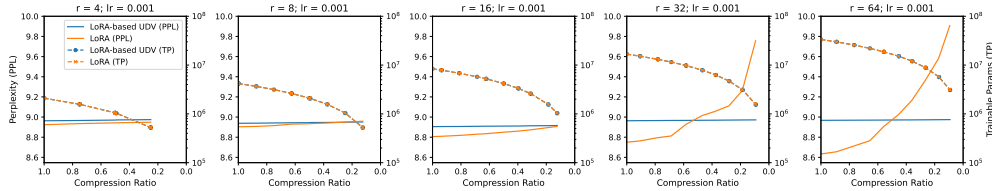


Figure SM26: Comparison of pruning performance between LoRA and LoRA-based UDV during training.

This toy example is not intended to present a mature pruning method, but rather to highlight the potential of the proposed UDV block, which can be applied to a wide range of models and demonstrates promising pruning performance.

## C Extended Discussion on the Related Work

**Burer-Monteiro factorization.** BM factorization was proposed for solving semidefinite programs (Burer & Monteiro, 2003, 2005; Boumal et al., 2016, 2020; Cifuentes, 2021) and has been recognized for its efficiency in addressing low-rank optimization problems, e.g., (Sun & Luo, 2016; Bhojanapalli et al., 2016; Park et al., 2017, 2018; Hsieh et al., 2018; Sahin et al., 2019; Lee et al., 2022; Yalcin et al., 2023). Building on the connections between training problems for (non-linear) two-layer neural networks and convex (copositive) formulations (see (Pilanci & Ergen, 2020; Ergen & Pilanci, 2020; Sahiner et al., 2021) and the references therein), Sahiner et al. (2023) recently introduced BM factorization to solve convex formulations of various architectures, including fully connected networks with ReLU and gated ReLU (Fiat et al., 2019) activations, two-layer convnets (LeCun et al., 1989a), and self-attention mechanisms based on Transformers (Vaswani et al., 2017).

**Implicit regularization.** One promising line of research that aims to explain the successful generalization abilities of neural networks follows the concept of ‘implicit regularization,’ which results from the optimization methods and formulations used during training (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). Le & Jegelka (2022) extend these results to the final linear layers of nonlinear ReLU-activated feedforward networks with fully connected layers and skip connections. More recently Timor et al. (2023) investigated implicit regularization in ReLU networks. Much of the existing work focuses on gradient flow dynamics in the limit of infinitesimal learning rates. In particular, Gidel et al. (2019) examined 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 (Goodfellow et al., 2016), 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 as hard constraints, addressing them within gradient-based optimization via their (sub)gradients. This approach is likely favored due to the ease of implementation, as pre-built functions are readily available in common neural network packages. Several independent studies have also applied proximal methods across different networks and applications, finding that proximal-based methods tend to yield solutions with more pronounced structures (Bai et al., 2019; Yang et al., 2020; Yun et al., 2021; Yurtsever et al., 2021; Yang et al., 2022). Structural regularization in the form of hard constraints, however, 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) 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.

There are other examples of constraints in neural networks. For instance, constraints can be applied to ensure adherence to real-world conditions in applications where they are necessary (Pathak et al., 2015; Márquez-Neila et al., 2017; Jia et al., 2017; Kervadec et al., 2019; Weber et al., 2021), or to incorporate physical laws in physics-informed neural networks (Raissi et al., 2019; Lu et al., 2021; Patel et al., 2022). In these cases, the feasible set is typically complex and difficult to project onto; therefore, optimization algorithms like primal-dual methods or augmented Lagrangian techniques are used. Constraints are also present in lifted neural networks, a framework where the training problem is reformulated in a higher-dimensional ‘lifted’ space. In this space, conventional activation functions like ReLU, used in the original formulation, are expressed as constraints (Askari et al., 2018; Sahiner et al., 2021; Bartan & Pilanci, 2021; Prakhya et al., 2024).

**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). Several compression techniques have been studied in the literature, including parameter quantization (Krishnamoorthi, 2018), knowledge distillation (Gou et al., 2021), and pruning.

Pruning reduces the number of parameters, resulting in more compact and efficient models that are easier to deploy. The literature on pruning is extensive, with diverse methods proposed with distinct characteristics. Various criteria are used to determine which weights to prune, including second-order derivative-based methods (LeCun et al., 1989b; Hassibi & Stork, 1992), magnitude-based pruning (Janowsky, 1989; Han et al., 2015), saliency heuristics (Mozer & Smolensky, 1988; Lee et al., 2018), and matrix or tensor factorization-based techniques (Xue et al., 2013; Sainath et al., 2013; Jaderberg et al., 2014; Lebedev et al., 2015; Swaminathan et al., 2020), among others. Pruning by singular value thresholding has 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). A comprehensive review of pruning methods is beyond the scope of this paper due to space limitations and the diversity of approaches. For more detailed reviews, we refer to (Reed, 1993; Blalock et al., 2020; Cheng et al., 2024), and the references therein.

Table SM2: Experiments using Adam optimizer. Not applicable or results with obvious oscillations or divergence are denoted as ‘-’.

Tasks Dataset (Transferred model)	Regression (Test Loss)		Classification (Test Accuracy)		
	HPART	NYCTTD	MNIST		
	( $\times 10^{-3}$ )	( $\times 10^{-6}$ )	(MaxVit-T [M]   EfficientNet-B0 [E]   RegNetX-32GF [R])	( $\times 100\%$ )	
LR: $10^{-6}$	-	-	UDV: - UDV-s: - UDV-v1: 99.56 UDV-v2: 99.53 UDV-ReLU: - UDV(unconstrained): 99.53 UFV: 99.53 UV-ReLU: 99.59 UV: 99.54 M: 99.42	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: 99.25 UV-ReLU: 99.29 UV: 99.24 E: 98.67	UDV: - UDV-s: - UDV-v1: 99.46 UDV-v2: 99.48 UDV-ReLU: - UDV(unconstrained): 99.48 UFV: 99.38 UV-ReLU: 99.41 UV: 99.36 R: 99.32
LR: $10^{-5}$	-	-	UDV: - UDV-s: - UDV-v1: 99.61 UDV-v2: 99.58 UDV-ReLU: - UDV(unconstrained): 99.62 UFV: 99.60 UV-ReLU: 99.57 UV: 99.61 M: 99.59	UDV: - UDV-s: - UDV-v1: 99.49 UDV-v2: 99.53 UDV-ReLU: - UDV(unconstrained): 99.54 UFV: 99.43 UV-ReLU: 99.52 UV: 99.52 E: 99.51	UDV: - UDV-s: - UDV-v1: <b>99.58</b> UDV-v2: 99.50 UDV-ReLU: - UDV(unconstrained): 99.50 UFV: 99.58 UV-ReLU: 99.59 UV: 99.57 R: <b>99.59</b>
LR: $10^{-4}$	UDV: 2.304 UDV-s: 1.912 UDV-v1: 1.823 UDV-v2: 1.738 UDV-ReLU: 2.731 UDV(unconstrained): 1.738 UFV: 1.351 UV-ReLU: 1.376 UV: 1.475	UDV: <b>5.248</b> UDV-s: 5.248 UDV-v1: <b>5.248</b> UDV-v2: 5.250 UDV-ReLU: <b>5.251</b> UDV(unconstrained): 5.250 UFV: 5.254 UV-ReLU: <b>5.263</b> UV: 5.275	UDV: <b>99.66</b> UDV-s: 99.58 UDV-v1: <b>99.65</b> UDV-v2: <b>99.69</b> UDV-ReLU: <b>99.63</b> UDV(unconstrained): <b>99.66</b> UFV: <b>99.69</b> UV-ReLU: <b>99.63</b> UV: <b>99.64</b> M: <b>99.65</b>	UDV: - UDV-s: - UDV-v1: <b>99.63</b> UDV-v2: <b>99.58</b> UDV-ReLU: - UDV(unconstrained): <b>99.63</b> UFV: <b>99.59</b> UV-ReLU: <b>99.54</b> UV: 99.56 E: 99.60	UDV: <b>99.55</b> UDV-s: <b>99.55</b> UDV-v1: 99.56 UDV-v2: <b>99.68</b> UDV-ReLU: <b>99.55</b> UDV(unconstrained): <b>99.59</b> UFV: <b>99.67</b> UV-ReLU: 99.64 UV: <b>99.60</b> R: 99.59
LR: $10^{-3}$	UDV: <b>1.304</b> UDV-s: <b>1.316</b> UDV-v1: <b>1.267</b> UDV-v2: <b>1.268</b> UDV-ReLU: <b>1.320</b> UDV(unconstrained): <b>1.268</b> UFV: <b>1.318</b> UV-ReLU: <b>1.167</b> UV: <b>1.333</b>	UDV: 5.248 UDV-s: <b>5.248</b> UDV-v1: 5.248 UDV-v2: <b>5.248</b> UDV-ReLU: 5.251 UDV(unconstrained): <b>5.248</b> UFV: <b>5.248</b> UV-ReLU: 5.306 UV: <b>5.251</b>	UDV: 99.57 UDV-s: <b>99.59</b> UDV-v1: 99.57 UDV-v2: 99.58 UDV-ReLU: 99.51 UDV(unconstrained): 99.52 UFV: 99.59 UV-ReLU: 99.60 UV: 99.60 M: 99.63	UDV: <b>99.55</b> UDV-s: <b>99.59</b> UDV-v1: 99.59 UDV-v2: 99.57 UDV-ReLU: <b>99.54</b> UDV(unconstrained): 99.57 UFV: 99.54 UV-ReLU: 99.49 UV: <b>99.60</b> E: <b>99.61</b>	UDV: 99.50 UDV-s: 99.49 UDV-v1: 99.47 UDV-v2: 99.47 UDV-ReLU: 99.42 UDV(unconstrained): 99.58 UFV: 99.41 UV-ReLU: <b>99.66</b> UV: 99.46 R: 99.49
LR: $10^{-2}$	UDV: 1.877 UDV-s: 1.998 UDV-v1: 1.409 UDV-v2: 1.500 UDV-ReLU: 1.699 UDV(unconstrained): 1.402 UFV: 1.483 UV-ReLU: 1.467 UV: 1.430	UDV: 5.257 UDV-s: 5.258 UDV-v1: 5.256 UDV-v2: 5.257 UDV-ReLU: 5.323 UDV(unconstrained): 5.263 UFV: 7.048 UV-ReLU: 5.323 UV: 7.369	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - M: -	UDV: 99.34 UDV-s: 99.56 UDV-v1: 99.48 UDV-v2: 99.44 UDV-ReLU: 96.95 UDV(unconstrained): 99.52 UFV: - UV-ReLU: 99.37 UV: 99.32 E: 98.42	UDV: 99.53 UDV-s: 99.43 UDV-v1: 99.35 UDV-v2: 99.28 UDV-ReLU: 99.37 UDV(unconstrained): 99.10 UFV: - UV-ReLU: 99.32 UV: 98.90 R: 99.39
LR: $10^{-1}$	UDV: 4.188 UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: 18.26 UDV(unconstrained): - UFV: 1.745 UV-ReLU: 42.01 UV: 1.614	UDV: 5.321 UDV-s: 114.6 UDV-v1: 23.12 UDV-v2: 21.22 UDV-ReLU: 5.323 UDV(unconstrained): 126.1 UFV: - UV-ReLU: 5.323 UV: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - M: -	UDV: 99.05 UDV-s: 95.62 UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: 97.69 E: 99.10	UDV: 97.54 UDV-s: 99.31 UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - R: 95.65
LR: $10^0$	UDV: 38.71 UDV-s: - UDV-v1: 2.413 UDV-v2: 4.633 UDV-ReLU: 48.62 UDV(unconstrained): 14.05 UFV: - UV-ReLU: 48.24 UV: -	UDV: 5.327 UDV-s: - UDV-v1: 16.19 UDV-v2: - UDV-ReLU: 5.323 UDV(unconstrained): - UFV: - UV-ReLU: 5.323 UV: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - M: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: -UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - E: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - R: 95.65
LR: $2 \times 10^0$	UDV: 60.46 UDV-s: - UDV-v1: 6.651 UDV-v2: 7.366 UDV-ReLU: 48.62 UDV(unconstrained): - UFV: - UV-ReLU: 49.43 UV: -	UDV: 5.322 UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: 5.323 UDV(unconstrained): - UFV: - UV-ReLU: 5.323 UV: -	-	-	-
LR: $3 \times 10^0$	UDV: 106.9 UDV-s: - UDV-v1: 6.606 UDV-v2: 7.398 UDV-ReLU: 48.62 UDV(unconstrained): - UFV: - UV-ReLU: 48.50 UV: -	UDV: 5.335 UDV-s: - UDV-v1: 6.380 UDV-v2: - UDV-ReLU: 5.323 UDV(unconstrained): - UFV: - UV-ReLU: 5.323 UV: -	-	-	-
LR: $5 \times 10^0$	-	-	-	-	-

Table SM3: Experiments using NAdam optimizer. Not applicable or results with obvious oscillations or divergence are denoted as ‘-’.

Tasks Dataset (Transferred model)	Regression (Test Loss)		Classification (Test Accuracy)		
	HPART	NYCTTD	MNIST		
	( $\times 10^{-3}$ )	( $\times 10^{-6}$ )	(MaxVit-T [M]   EfficientNet-B0 [E]   RegNetX-32GF [R])	( $\times 100\%$ )	
LR: $10^{-6}$	-	-	UDV: - UDV-s: - UDV-v1: 99.56 UDV-v2: 99.53 UDV-ReLU: - UDV(unconstrained): 99.53 UFV: 99.54 UV-ReLU: 99.59 UV: 99.53 M: 99.41	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: 99.26 UV-ReLU: 99.30 UV: 99.24 E: 98.67	UDV: - UDV-s: - UDV-v1: 99.49 UDV-v2: 99.46 UDV-ReLU: - UDV(unconstrained): 99.46 UFV: 99.32 UV-ReLU: 99.40 UV: 99.32 R: 99.37
LR: $10^{-5}$	-	-	UDV: - UDV-s: - UDV-v1: 99.63 UDV-v2: 99.59 UDV-ReLU: - UDV(unconstrained): 99.60 UFV: 99.62 UV-ReLU: 99.61 UV: 99.63 M: 99.62	UDV: - UDV-s: - UDV-v1: 99.50 UDV-v2: 99.52 UDV-ReLU: - UDV(unconstrained): 99.52 UFV: 99.43 UV-ReLU: 99.51 UV: 99.52 E: 99.50	UDV: - UDV-s: - UDV-v1: 99.66 UDV-v2: 99.56 UDV-ReLU: - UDV(unconstrained): 99.56 UFV: 99.54 UV-ReLU: 99.59 UV: 99.61 R: 99.67
LR: $10^{-4}$	UDV: 2.312 UDV-s: 1.916 UDV-v1: 1.837 UDV-v2: 1.752 UDV-ReLU: 2.665 UDV(unconstrained): 1.752 UFV: 1.381 UV-ReLU: 1.398 UV: 1.512	UDV: 5.248 UDV-s: 5.248 UDV-v1: 5.248 UDV-v2: 5.249 UDV-ReLU: 5.251 UDV(unconstrained): 5.249 UFV: 5.256 UV-ReLU: 5.258 UV: 5.275	UDV: 99.67 UDV-s: 99.62 UDV-v1: 99.67 UDV-v2: 99.61 UDV-ReLU: 99.63 UDV(unconstrained): 99.62 UFV: 99.68 UV-ReLU: 99.68 UV: 99.65 M: 99.59	UDV: - UDV-s: - UDV-v1: 99.60 UDV-v2: 99.66 UDV-ReLU: - UDV(unconstrained): 99.60 UFV: 99.54 UV-ReLU: 99.53 UV: 99.53 E: 99.65	UDV: 99.52 UDV-s: 99.55 UDV-v1: 99.69 UDV-v2: 99.72 UDV-ReLU: 99.46 UDV(unconstrained): 99.63 UFV: 99.55 UV-ReLU: 99.63 UV: 99.64 R: 99.64
LR: $10^{-3}$	UDV: 1.638 UDV-s: 1.691 UDV-v1: 1.418 UDV-v2: 1.440 UDV-ReLU: 1.504 UDV(unconstrained): 1.437 UFV: 1.607 UV-ReLU: 1.367 UV: 1.654	UDV: 5.248 UDV-s: 5.248 UDV-v1: 5.248 UDV-v2: 5.248 UDV-ReLU: 5.251 UDV(unconstrained): 5.248 UFV: 5.248 UV-ReLU: 5.249 UV: 5.254	UDV: 99.53 UDV-s: 99.56 UDV-v1: 99.54 UDV-v2: 99.61 UDV-ReLU: 99.55 UDV(unconstrained): 99.66 UFV: 99.57 UV-ReLU: 99.63 UV: 99.59 M: 99.58	UDV: 97.56 UDV-s: 99.61 UDV-v1: 99.60 UDV-v2: 99.55 UDV-ReLU: 99.50 UDV(unconstrained): 99.55 UFV: 99.28 UV-ReLU: 99.55 UV: 99.55 E: 99.58	UDV: 99.45 UDV-s: 99.45 UDV-v1: 99.43 UDV-v2: 99.46 UDV-ReLU: 99.47 UDV(unconstrained): 99.50 UFV: 99.27 UV-ReLU: 99.53 UV: 99.42 R: 99.65
LR: $10^{-2}$	UDV: 3.396 UDV-s: 3.287 UDV-v1: 2.297 UDV-v2: 2.315 UDV-ReLU: 2.403 UDV(unconstrained): 1.884 UFV: - UV-ReLU: 1.954 UV: -	UDV: 5.258 UDV-s: 6.918 UDV-v1: 5.276 UDV-v2: 5.253 UDV-ReLU: 5.323 UDV(unconstrained): 5.253 UFV: 25.24 UV-ReLU: 5.323 UV: 6.262	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - M: -	UDV: 99.35 UDV-s: - UDV-v1: 99.31 UDV-v2: 99.01 UDV-ReLU: 99.42 UDV(unconstrained): 98.78 UFV: - UV-ReLU: 98.87 UV: 98.60 E: 99.26	UDV: 97.12 UDV-s: 99.28 UDV-v1: 99.37 UDV-v2: 99.38 UDV-ReLU: 99.49 UDV(unconstrained): 99.38 UFV: - UV-ReLU: 99.28 UV: 98.91 R: 99.59
LR: $10^{-1}$	UDV: 13.69 UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: 48.62 UDV(unconstrained): - UFV: 4.918 UV-ReLU: 41.61 UV: 1.863	UDV: 5.323 UDV-s: - UDV-v1: 6.627 UDV-v2: - UDV-ReLU: 5.323 UDV(unconstrained): - UFV: - UV-ReLU: 5.323 UV: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - M: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: 97.54 E: 98.75	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - R: 99.19
LR: $10^0$	UDV: - UDV-s: - UDV-v1: 5.649 UDV-v2: 2.317 UDV-ReLU: 48.62 UDV(unconstrained): - UFV: - UV-ReLU: 45.77 UV: -	UDV: 5.328 UDV-s: 53.70 UDV-v1: 19.52 UDV-v2: - UDV-ReLU: 5.323 UDV(unconstrained): - UFV: - UV-ReLU: 5.323 UV: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - M: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - E: -	UDV: - UDV-s: - UDV-v1: - UDV-v2: - UDV-ReLU: - UDV(unconstrained): - UFV: - UV-ReLU: - UV: - R: -
LR: $2 \times 10^0$	UDV: - UDV-s: - UDV-v1: 3.075 UDV-v2: 2.828 UDV-ReLU: 48.62 UDV(unconstrained): - UFV: - UV-ReLU: 59.82 UV: -	UDV: 5.377 UDV-s: - UDV-v1: 6.011 UDV-v2: - UDV-ReLU: 5.323 UDV(unconstrained): - UFV: - UV-ReLU: 5.323 UV: -	-	-	-
LR: $3 \times 10^0$	UDV: - UDV-s: - UDV-v1: 3.882 UDV-v2: 3.128 UDV-ReLU: 48.62 UDV(unconstrained): - UFV: - UV-ReLU: - UV: -	UDV: 5.401 UDV-s: - UDV-v1: 15.27 UDV-v2: - UDV-ReLU: 5.323 UDV(unconstrained): - UFV: - UV-ReLU: 5.323 UV: -	-	-	-
LR: $5 \times 10^0$	-	-	-	-	-

Table SM4: Experiments using MBGD optimizer. Not applicable or results with obvious oscillations or divergence are denoted as ‘-’.

Tasks		Regression (Test Loss)		Classification (Test Accuracy)		
Dataset		HPART	NYCTTD	MNIST		
(Transferred model)		( $\times 10^{-3}$ )	( $\times 10^{-6}$ )	(MaxVit-T [M]   EfficientNet-B0 [E]   RegNetX-32GF [R])	(×100%)	
LR: $10^{-6}$		-	-	-	-	-
LR: $10^{-5}$		-	-	-	-	-
LR: $10^{-4}$	UDV: 46.90	UDV: 26.60				
	UDV-s: 45.43	UDV-s: 40.06				
	UDV-v1: 46.01	UDV-v1: 49.35				
	UDV-v2: 44.35	UDV-v2: 83.91				
	UDV-ReLU: 47.88	UDV-ReLU: 20.56				
	UDV(unconstrained): 44.35	UDV(unconstrained): 83.91				
	UFV: 11.77	UFV: 86.68				
LR: $10^{-3}$	UV-ReLU: 12.59	UV-ReLU: -				
	UV: 11.98	UV: 71.72				
	UDV: 30.38	UDV: 9.407	UDV: -	UDV: -	UDV: -	UDV: -
	UDV-s: 21.39	UDV-s: 10.29	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -
	UDV-v1: 23.65	UDV-v1: 10.83	UDV-v1: 98.44	UDV-v1: -	UDV-v1: -	UDV-v1: -
	UDV-v2: 15.77	UDV-v2: 12.56	UDV-v2: 98.32	UDV-v2: -	UDV-v2: -	UDV-v2: -
	UDV-ReLU: 39.89	UDV-ReLU: 7.231	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -
LR: $10^{-2}$	UDV(unconstrained): 15.77	UDV(unconstrained): 12.56	UDV(unconstrained): 98.27	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -
	UFV: 6.719	UFV: 8.236	UFV: 99.34	UFV: 98.15	UFV: 99.20	UFV: 99.20
	UV-ReLU: 6.478	UV-ReLU: 12.67	UV-ReLU: 99.24	UV-ReLU: 98.48	UV-ReLU: 99.16	UV-ReLU: 99.16
	UV: 6.345	UV: 6.763	UV: 99.19	UV: 98.55	UV: 99.17	UV: 99.17
			M: 99.29	E: 98.74	R: 99.16	
	UDV: 6.030	UDV: 5.465	UDV: 99.38	UDV: -	UDV: 99.40	UDV: 99.40
	UDV-s: 6.014	UDV-s: 5.421	UDV-s: 99.32	UDV-s: -	UDV-s: 99.46	UDV-s: 99.46
LR: $10^{-1}$	UDV-v1: 6.125	UDV-v1: 5.514	UDV-v1: 99.47	UDV-v1: 98.62	UDV-v1: 99.37	UDV-v1: 99.37
	UDV-v2: 5.877	UDV-v2: 5.532	UDV-v2: 99.50	UDV-v2: 98.97	UDV-v2: 99.34	UDV-v2: 99.34
	UDV-ReLU: 6.234	UDV-ReLU: 5.431	UDV-ReLU: 99.53	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -
	UDV(unconstrained): 5.877	UDV(unconstrained): 5.532	UDV(unconstrained): 99.58	UDV(unconstrained): 98.40	UDV(unconstrained): 99.34	UDV(unconstrained): 99.34
	UFV: 5.575	UFV: 5.293	UFV: 99.53	UFV: 99.05	UFV: 99.36	UFV: 99.36
	UV-ReLU: 2.253	UV-ReLU: 5.521	UV-ReLU: 99.53	UV-ReLU: 99.31	UV-ReLU: 99.31	UV-ReLU: 99.31
	UV: 2.251	UV: 5.296	UV: 99.50	UV: 99.24	UV: 99.38	UV: 99.38
LR: $10^0$			M: 99.59	E: 99.43	R: 99.44	
	UDV: <b>1.398</b>	UDV: 5.253	UDV: <b>99.59</b>	UDV: -	UDV: <b>99.51</b>	UDV: <b>99.51</b>
	UDV-s: <b>1.407</b>	UDV-s: 5.250	UDV-s: <b>99.38</b>	UDV-s: <b>99.36</b>	UDV-s: <b>99.57</b>	UDV-s: <b>99.57</b>
	UDV-v1: <b>1.556</b>	UDV-v1: 5.256	UDV-v1: <b>99.61</b>	UDV-v1: 99.43	UDV-v1: 99.60	UDV-v1: 99.60
	UDV-v2: <b>1.565</b>	UDV-v2: 5.252	UDV-v2: <b>99.60</b>	UDV-v2: 99.53	UDV-v2: 99.62	UDV-v2: 99.62
	UDV-ReLU: <b>1.379</b>	UDV-ReLU: 5.277	UDV-ReLU: <b>99.65</b>	UDV-ReLU: 95.21	UDV-ReLU: -	UDV-ReLU: -
	UDV(unconstrained): <b>1.565</b>	UDV(unconstrained): 5.252	UDV(unconstrained): <b>99.59</b>	UDV(unconstrained): <b>95.29</b>	UDV(unconstrained): <b>99.61</b>	UDV(unconstrained): <b>99.61</b>
LR: $10^1$	UFV: <b>1.548</b>	UFV: <b>5.255</b>	UFV: <b>99.61</b>	UFV: <b>99.55</b>	UFV: <b>99.59</b>	UFV: <b>99.59</b>
	UV-ReLU: <b>1.493</b>	UV-ReLU: <b>5.323</b>	UV-ReLU: <b>99.60</b>	UV-ReLU: 99.52	UV-ReLU: 99.59	UV-ReLU: 99.59
	UV: <b>1.583</b>	UV: <b>5.291</b>	UV: <b>99.63</b>	UV: <b>99.55</b>	UV: 98.56	UV: 98.56
			M: <b>99.62</b>	E: <b>99.67</b>	R: 99.63	
	UDV: 3.076	UDV: <b>5.248</b>	UDV: -	UDV: -	UDV: -	UDV: -
	UDV-s: 2.870	UDV-s: <b>5.248</b>	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -
	UDV-v1: 1.935	UDV-v1: <b>5.249</b>	UDV-v1: -	UDV-v1: <b>99.65</b>	UDV-v1: <b>99.67</b>	UDV-v1: <b>99.67</b>
LR: $2 \times 10^0$	UDV-v2: 1.830	UDV-v2: <b>5.248</b>	UDV-v2: -	UDV-v2: 99.59	UDV-v2: <b>99.69</b>	UDV-v2: <b>99.69</b>
	UDV-ReLU: 4.573	UDV-ReLU: 5.261	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -
	UDV(unconstrained): 1.810	UDV(unconstrained): <b>5.248</b>	UDV(unconstrained): -	UDV(unconstrained): <b>99.62</b>	UDV(unconstrained): 99.57	UDV(unconstrained): 99.57
	UFV: -	UFV: 5.256	UFV: -	UFV: -	UFV: -	UFV: -
	UV-ReLU: 48.62	UV-ReLU: 5.271	UV-ReLU: -	UV-ReLU: <b>99.64</b>	UV-ReLU: <b>99.73</b>	UV-ReLU: <b>99.73</b>
	UV: -	UV: -	UV: -	UV: -	UV: <b>99.66</b>	UV: <b>99.66</b>
			M: -	E: 99.55	R: <b>99.66</b>	
LR: $3 \times 10^0$	UDV: 6.244	UDV: 5.248	UDV: -	UDV: -	UDV: -	UDV: -
	UDV-s: 48.63	UDV-s: 5.248	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -
	UDV-v1: 2.559	UDV-v1: 5.249	UDV-v1: -	UDV-v1: 98.37	UDV-v1: 99.66	UDV-v1: 99.66
	UDV-v2: -	UDV-v2: 5.249	UDV-v2: -	UDV-v2: <b>99.66</b>	UDV-v2: 99.55	UDV-v2: 99.55
	UDV-ReLU: 47.50	UDV-ReLU: 5.259	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -
	UDV(unconstrained): -	UDV(unconstrained): 5.249	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -
	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -
LR: $5 \times 10^0$	UV-ReLU: 48.63	UV-ReLU: 5.293	UV-ReLU: -	UV-ReLU: -	UV-ReLU: 99.58	UV-ReLU: 99.58
	UV: -	UV: -	UV: -	UV: -	UV: 99.57	UV: 99.57
			M: -	E: 98.88	R: 99.56	
	UDV: -	UDV: 5.248	UDV: -	UDV: -	UDV: -	UDV: -
	UDV-s: -	UDV-s: 5.248	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -
	UDV-v1: -	UDV-v1: 5.249	UDV-v1: -	UDV-v1: 99.07	UDV-v1: 99.48	UDV-v1: 99.48
	UDV-v2: -	UDV-v2: 5.249	UDV-v2: -	UDV-v2: -	UDV-v2: 99.65	UDV-v2: 99.65
LR: $10^2$	UDV-ReLU: 48.62	UDV-ReLU: <b>5.257</b>	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -
	UDV(unconstrained): -	UDV(unconstrained): 5.250	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -
	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -
	UV-ReLU: 48.62	UV-ReLU: 5.296	UV-ReLU: -	UV-ReLU: -	UV-ReLU: 99.57	UV-ReLU: 99.57
	UV: -	UV: -	UV: -	UV: -	UV: -	UV: -
			M: -	E: 99.26	R: 99.57	
LR: $10^3$	UDV: -	UDV: -	UDV: -	UDV: -	UDV: -	UDV: -
	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -
	UDV-v1: -	UDV-v1: -	UDV-v1: -	UDV-v1: -	UDV-v1: 99.55	UDV-v1: 99.55
	UDV-v2: -	UDV-v2: -	UDV-v2: -	UDV-v2: -	UDV-v2: 99.41	UDV-v2: 99.41
	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -
	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -
	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -
LR: $10^4$	UV-ReLU: -	UV-ReLU: -	UV-ReLU: -	UV-ReLU: -	UV-ReLU: 99.42	UV-ReLU: 99.42
	UV: -	UV: -	UV: -	UV: -	UV: -	UV: -
			M: -	E: 99.33	R: 99.55	

Table SM5: Experiments using MBGDM optimizer. Not applicable or results with obvious oscillations or divergence are denoted as ‘-’.

Tasks		Regression (Test Loss)		Classification (Test Accuracy)		
Dataset		HPART	NYCTTD	MNIST		
(Transferred model)		( $\times 10^{-3}$ )	( $\times 10^{-6}$ )	(MaxVit-T [M]   EfficientNet-B0 [E]   RegNetX-32GF [R])		
				( $\times 100\%$ )		
LR: $10^{-6}$		-	-	-	-	-
LR: $10^{-5}$		-	-	-	-	-
LR: $10^{-4}$	UDV: 30.52	UDV: 9.413				
	UDV-s: 21.50	UDV-s: 10.29				
	UDV-v1: 23.79	UDV-v1: 10.85				
	UDV-v2: 15.84	UDV-v2: 12.59				
	UDV-ReLU: 40.00	UDV-ReLU: 7.224				
	UDV(unconstrained): 15.84	UDV(unconstrained): 12.59				
	UFV: 6.729	UFV: 8.288				
LR: $10^{-3}$	UV-ReLU: 6.488	UV-ReLU: 12.68				
	UV: 6.348	UV: 6.776				
	UDV: 6.105	UDV: 5.457	UDV: 99.26	UDV: -	UDV: 99.50	
	UDV-s: 6.080	UDV-s: 5.412	UDV-s: 99.38	UDV-s: -	UDV-s: 99.43	
	UDV-v1: 6.178	UDV-v1: 5.516	UDV-v1: 99.39	UDV-v1: 99.00	UDV-v1: 99.39	
	UDV-v2: 5.929	UDV-v2: 5.534	UDV-v2: 99.52	UDV-v2: 99.05	UDV-v2: 99.37	
	UDV-ReLU: 6.341	UDV-ReLU: 5.431	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	
LR: $10^{-2}$	UDV(unconstrained): 5.929	UDV(unconstrained): 5.534	UDV(unconstrained): 99.51	UDV(unconstrained): 99.08	UDV(unconstrained): 99.40	
	UFV: 2.590	UFV: 5.294	UFV: 99.56	UFV: 99.18	UFV: 99.38	
	UV-ReLU: 2.529	UV-ReLU: 5.520	UV-ReLU: 99.46	UV-ReLU: 99.37	UV-ReLU: 99.36	
	UV: 2.252	UV: 5.296	UV: 99.54	UV: 99.36	UV: 99.30	
			M: 99.58	E: 99.45	R: 99.41	
	UDV: 1.357	UDV: 5.253	UDV: <b>99.67</b>	UDV: 99.48	UDV: <b>99.74</b>	
	UDV-s: 1.388	UDV-s: 5.250	UDV-s: 99.38	UDV-s: 99.60	UDV-s: <b>99.72</b>	
LR: $10^{-1}$	UDV-v1: 1.554	UDV-v1: 5.256	UDV-v1: 99.63	UDV-v1: 99.54	UDV-v1: 99.61	
	UDV-v2: 1.569	UDV-v2: 5.252	UDV-v2: 99.60	UDV-v2: 99.59	UDV-v2: <b>99.67</b>	
	UDV-ReLU: <b>1.312</b>	UDV-ReLU: 5.276	UDV-ReLU: <b>99.62</b>	UDV-ReLU: 99.54	UDV-ReLU: <b>99.65</b>	
	UDV(unconstrained): 1.569	UDV(unconstrained): 5.252	UDV(unconstrained): 99.59	UDV(unconstrained): 99.55	UDV(unconstrained): 99.63	
	UFV: <b>1.357</b>	UFV: 5.254	UFV: <b>99.65</b>	UFV: 99.50	UFV: <b>99.66</b>	
	UV-ReLU: 1.314	UV-ReLU: 5.267	UV-ReLU: <b>99.63</b>	UV-ReLU: 99.53	UV-ReLU: 99.61	
	UV: <b>1.337</b>	UV: 5.279	UV: <b>99.69</b>	UV: 99.50	UV: 99.53	
LR: $10^0$			M: <b>99.64</b>	E: 99.59	R: 99.60	
	UDV: <b>1.345</b>	UDV: <b>5.248</b>	UDV: 99.61	UDV: <b>99.63</b>	UDV: 99.60	
	UDV-s: <b>1.339</b>	UDV-s: 5.248	UDV-s: <b>99.60</b>	UDV-s: <b>99.63</b>	UDV-s: 99.57	
	UDV-v1: <b>1.313</b>	UDV-v1: 5.249	UDV-v1: <b>99.65</b>	UDV-v1: <b>99.64</b>	UDV-v1: <b>99.71</b>	
	UDV-v2: <b>1.302</b>	UDV-v2: 5.248	UDV-v2: <b>99.68</b>	UDV-v2: <b>99.66</b>	UDV-v2: 99.67	
	UDV-ReLU: 1.318	UDV-ReLU: 5.259	UDV-ReLU: -	UDV-ReLU: <b>99.61</b>	UDV-ReLU: -	
	UDV(unconstrained): <b>1.302</b>	UDV(unconstrained): 2.248	UDV(unconstrained): <b>99.70</b>	UDV(unconstrained): <b>99.59</b>	UDV(unconstrained): <b>99.67</b>	
LR: $2 \times 10^0$	UFV: 1.358	UFV: <b>5.251</b>	UFV: -	UFV: <b>99.68</b>	UFV: 99.63	
	UV-ReLU: <b>1.244</b>	UV-ReLU: <b>5.264</b>	UV-ReLU: 99.63	UV-ReLU: <b>99.68</b>	UV-ReLU: <b>99.66</b>	
	UV: -	UV: <b>5.259</b>	UV: -	UV: <b>99.59</b>	UV: <b>99.56</b>	
			M: -	E: <b>99.63</b>	R: <b>99.67</b>	
	UDV: 123.6	UDV: 5.248	UDV: -	UDV: -	UDV: -	
	UDV-s: -	UDV-s: 5.248	UDV-s: -	UDV-s: -	UDV-s: -	
	UDV-v1: -	UDV-v1: 5.249	UDV-v1: -	UDV-v1: 99.54	UDV-v1: 99.52	
LR: $3 \times 10^0$	UDV-v2: -	UDV-v2: 5.249	UDV-v2: -	UDV-v2: -	UDV-v2: 99.58	
	UDV-ReLU: 47.03	UDV-ReLU: 5.251	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	
	UDV(unconstrained): -	UDV(unconstrained): 5.249	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	
	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -	
	UV-ReLU: 48.62	UV-ReLU: 5.281	UV-ReLU: -	UV-ReLU: -	UV-ReLU: 99.53	
	UV: -	UV: -	UV: -	UV: -	UV: -	
			M: -	E: 99.24	R: 99.34	
LR: $5 \times 10^0$	UDV: -	UDV: 5.248	UDV: -	UDV: -	UDV: -	
	UDV-s: -	UDV-s: 5.248	UDV-s: -	UDV-s: -	UDV-s: -	
	UDV-v1: -	UDV-v1: 5.249	UDV-v1: -	UDV-v1: -	UDV-v1: 99.31	
	UDV-v2: -	UDV-v2: 5.248	UDV-v2: -	UDV-v2: -	UDV-v2: 99.37	
	UDV-ReLU: 48.62	UDV-ReLU: <b>5.249</b>	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	
	UDV(unconstrained): -	UDV(unconstrained): 5.248	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	
	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -	
LR: $10^{-6}$	UV-ReLU: 48.62	UV-ReLU: 5.289	UV-ReLU: -	UV-ReLU: -	UV-ReLU: 99.28	
	UV: -	UV: -	UV: -	UV: -	UV: -	
			M: -	E: 98.95	R: 99.31	
	UDV: -	UDV: 5.248	UDV: -	UDV: -	UDV: -	
	UDV-s: -	UDV-s: <b>5.248</b>	UDV-s: -	UDV-s: -	UDV-s: -	
	UDV-v1: -	UDV-v1: <b>5.248</b>	UDV-v1: -	UDV-v1: -	UDV-v1: -	
	UDV-v2: -	UDV-v2: <b>5.248</b>	UDV-v2: -	UDV-v2: -	UDV-v2: -	
LR: $10^{-5}$	UDV-ReLU: 48.62	UDV-ReLU: 5.249	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	
	UDV(unconstrained): -	UDV(unconstrained): <b>5.248</b>	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	
	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -	
	UV-ReLU: 48.62	UV-ReLU: 5.292	UV-ReLU: -	UV-ReLU: -	UV-ReLU: -	
	UV: -	UV: -	UV: -	UV: -	UV: -	
			M: -	E: 98.70	R: 99.12	
LR: $10^{-4}$	UDV: -	UDV: -	UDV: -	UDV: -	UDV: -	
	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -	UDV-s: -	
	UDV-v1: -	UDV-v1: -	UDV-v1: -	UDV-v1: -	UDV-v1: 99.55	
	UDV-v2: -	UDV-v2: -	UDV-v2: -	UDV-v2: -	UDV-v2: 99.41	
	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	UDV-ReLU: -	
	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	UDV(unconstrained): -	
	UFV: -	UFV: -	UFV: -	UFV: -	UFV: -	
LR: $10^{-3}$	UV-ReLU: -	UV-ReLU: -	UV-ReLU: -	UV-ReLU: -	UV-ReLU: 99.42	
	UV: -	UV: -	UV: -	UV: -	UV: -	
			M: -	E: -	R: 99.55	



## 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](https://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.