# Fast and Differentiable Matrix Inverse and Its Extension to SVD

**Anonymous authors**
Paper under double-blind review

## Abstract

Matrix inverse (Minv) and singular value decomposition (SVD) are among the most widely used matrix operations in massive data analysis, machine learning, and statistics. Although well-studied, they still encounter difficulties in practical use due to inefficiency and non-differentiability. In this paper, we aim at solving efficiency and differentiability issues through learning-based methods. First of all, to perform matrix inverse, we provide a differentiable yet efficient way, named LD-Minv, which is a learnable deep neural network (DNN) with each layer being an $L$-th order matrix polynomial. We show that, with proper initialization, the difference between LD-Minv's output and exact pseudo-inverse is in the order $\mathcal{O}\big(\exp\big\{-L^K\big\}\big)$, where $K$ is the depth of the LD-Minv. Moreover, by learning from data, LD-Minv further reduces the difference between the output and the exact pseudo-inverse. We prove that gradient descent finds an $\epsilon$-error minimum within $\mathcal{O}(nKL\log 1/\epsilon)$ steps for LD-Minv, where $n$ is the data size. At last, we provide the generalization bound for LD-Minv in both under-parameterized and over-parameterized settings. As an application of LD-Minv, we provide a learning-based optimization method to solve the problem with orthogonality constraints and utilize it to differentiate SVD (D-SVD). We also provide a theoretical generalization guarantee for D-SVD. Finally, we demonstrate the superiority of our methods on the synthetic and real data in the supplementary materials.

## 1 Introduction

Matrix inverse (including matrix pseudo-inverse) and singular value decomposition are fundamental linear algebra operations, ubiquitous in machine learning, statistics, signal processing, and other fields. In general, solving scientific computing or optimization problems often need to perform these two operators, such as matrix (pseudo) inverse for least squares regression, singular value decomposition (SVD) for dimensionality reduction (PCA), the low-rank related problems (Liu et al., 2010; Zhang et al., 2018; Liu et al., 2013), the graph-based issues (Wu et al., 2020a; Von Luxburg, 2007), and even for training deep neural networks (DNNs) with structured layers (Ionescu et al., 2015). Nevertheless, Minv and SVD appear less and less often in the modern machine learning tasks. One reason is inefficiency. Computing the SVD and the Minv can be extremely time-consuming for large-scale problems; however, efficiency is a significant concern in the current big data and deep learning era. Besides, non-differentiability is considered another reason that blocks the use of SVD and Minv. Usually, most prevalent methods for training DNNs are first-order and are based on the backpropagation; however, Minv and SVD are not necessarily continuous functions of the matrix entries (Stewart, 1969). Therefore, derivatives are not always existent. Although Minv and SVD may be backprop-able due to some specific implementation, they are unstable and are essentially non-differentiable. Thus the gradients cannot pass through them when backpropagating. Considering the above problems, one natural question emerges.

*Does there exist an efficient and differentiable way to perform Minv and SVD?*

Over the last decade, many sketches-based methods have been developed, e.g., Nelson & Nguyen (2013); Meng & Mahoney (2013); Cohen et al. (2015); Indyk et al. (2019). The main idea of the sketches-based techniques is to use random projections, which is efficiently computable, to reduce the problem size before performing SVD and Minv. However, they do not solve the problem of non-differentiability as smaller-sized SVD and Minv still needs to be computed.

Recently, "differentiable learning-based" methods (D-LbM) have attracted considerable attention (Chen et al., 2018; Indyk et al., 2019; Liu et al., 2019; Wu et al., 2020b; Xie et al., 2019). These methods usually unroll the classical optimization or numerical iterative algorithms and introduce the learnable parameters to obtain a learnable DNN. In general, the iterative algorithms inspired DNNs consist of differentiable operators such as matrix polynomials. Benefiting from training on the data, D-LbMs can execute in much fewer iterations with similar per-iteration cost as original algorithms but obtain much better performance. Many empirical results, e.g., Gregor & LeCun (2010); Yang et al. (2016); Peng et al. (2018), show that a well-trained DNN provided by D-LbM—compared with the original optimization algorithm—can obtain an almost equally good solution using one or even two order-of-magnitude fewer iterations. Based on these observations, we aim to find a learning-based iterative way to differentiate Minv and SVD in this paper.

However, all these D-LbMs suffer from two common problems. One is the convergence guarantee. The forward process of D-LbMs may diverge, even if the original unrolled iterative algorithm has a well-behaved convergence guarantee. In fact, most D-LbM methods have little or no convergence guarantees. To the best of our knowledge, there is no work to reveal the behavior of D-LbM during training. How does the training loss decrease? How big is the gap between the output of D-LbM and that of the original algorithm? All these questions are still open. Another problem is, both for D-LbM and the original algorithm, a limited theory exists when the input data obey a common distribution (e.g., data drawn from a low-dimensional manifold). Essentially, by learning from data, D-LbMs can obtain a problem-dependent parameter bias. It can help D-LbMs get the better performance on a specific data distribution within much fewer computation cost, instead of simply fixing the parameter ahead of time like traditional iterative methods. But there is no mathematical result to describe this phenomenon strictly. Moreover, it is unknown whether the trained D-LbMs generalize well on data from the same distribution.

Remarkably, in this paper we provide a learnable, differentiable and efficient way to perform Minv, named *Learnable Differentiable Matrix Inverse* (LD-Minv) and solve the above two problems by our proposed LD-Minv. First of all, LD-Minv is a DNN with each layer being an $L$-th order matrix polynomial; and the coefficient of each power is learnable. We show that LD-Minv converges to the Minv or pseudo-inverse with order $L$ if the coefficients of polynomial are set properly. Namely, the difference between the output of the DNN and the exact pseudo-inverse is in the order $\mathcal{O}\big(\exp\{-L^K\}\big)$, where $K$ is the depth of LD-Minv. Secondly, by learning from data, LD-Minv can further improve the and precision on some specific data distribution. Namely, the distance between the output and the exact pseudo-inverse can be arbitrarily small. Specifically, the training loss converges to zero exponentially fast, i.e., gradient descent (GD) finds an $\epsilon$-error minimum in $\ell_2$ regression using at most $\mathcal{O}(nKL\log 1/\epsilon)$ iterations, where $n$ is the data size. Finally, we provide the generalization bound for LD-Minv in both under-parameterized and over-parameterized settings.

With LD-Minv at hand, as a direct application, we propose a learning-based optimization to solve any convex problems with non-convex orthogonality constraints. Then we use it to differentiate SVD. Note that we also provide a generalization guarantee for our D-SVD. Unlike the previous work on differentiable SVD (Indyk et al., 2019), which is based on the power method and needs an assumption on the gap between singular values to ensure convergence, our method can converge to the solution without any gap assumption. In summary, our main contributions include:

- We propose a differentiable method, LD-Minv, to perform matrix inverse efficiently. LD-Minv, inspiring a DNN with each layer being a learnable $L$-th order matrix polynomial, would be useful for accelerating and differentiating general matrix decompositions. We show that LD-Minv can converge to the matrix pseudo-inverse with order $L$.
- By learning, LD-Minv can further improve the approximation performance on some underlying data distribution. We prove that GD finds an $\epsilon$-error minimum in $\ell_2$ regression using at most $\mathcal{O}(nKL\log 1/\epsilon)$ iterations under mild conditions. Moreover, we also provide the generalization bound for LD-Minv. We reveal that the empirical Rademacher complexity of the loss function class is bounded by $\widetilde{\mathcal{O}}(\min\{d^3L/K^{1/2}, \sqrt{d^3K/n}\})$, where $\widetilde{\mathcal{O}}(\cdot)$ hides log factors, and $K$ and $d$ are the depth and the width of LD-Minv, respectively.
- As a direct application, we further provide a learning-based general framework to solve the problems with orthogonality constraints. This D-LbM helps us to differentiate SVD. Finally, we also provide a generalization guarantee for our D-SVD.

## 2 Differentiable Matrix Inverse

We introduce the proposed D-Minv in this section. We first present an intuitive idea to show how to approximate the matrix inverse iteratively. Then we generalize the iterative method to obtain our fixed and learnable D-Minv, separately

### 2.1 Fixed Differentiable Matrix Inverse

Given a matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, we want to approximate its inverse $\mathbf{A}^{-1}$ by the matrix $\mathbf{X} \in \mathbb{R}^{d\times d}$, i.e., $\mathbf{AX} \approx \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix. Ideally, $\mathbf{X}$ is the fixed point of the following equation:

$$\mathbf{X} = \mathbf{X}(\mathbf{AX})^{-1} = \mathbf{X}\left(\sum_{l=0}^{\infty}(\mathbf{I} - \mathbf{AX})^l\right), \tag{1}$$

where the last equality follows from the Neumann series when $\mathbf{AX} \approx \mathbf{I}$ and $(\mathbf{I} - \mathbf{AX})^l$ is the $l$-order matrix polynomial.

Inspired by the above iterative process, it is obvious that we can obtain the matrix $\mathbf{X}$ iteratively. Hence, we consider the following higher-order iterative method, which is an $L$-th order matrix polynomial in each iteration, where $L \geq 4$:

$$\mathbf{X}_{k+1} = \mathbf{X}_k\left(\sum_{l=0}^{L-1}\mathbf{E}_k^l\right), \quad \mathbf{E}_k = \mathbf{I} - \mathbf{AX}_k. \tag{2}$$

Note that DNNs can easily implement this method: one layer corresponds to one iterative step. Since there are no parameters to learn in Eq. (2), and it only involves the matrix multiplication (thus differentiable), we name it fixed D-Minv [1]. One may doubt that the computational cost of each iteration is high in Eq. (2). However, as shown in Lemma 1, higher-order polynomial usually implies a faster convergence rate. For obtaining the same precision with different $L$, the total computational cost is in the order of $\mathcal{O}(L/\ln(L))$, which is a very slow growth rate w.r.t. $L$. Although simple, fixed D-Minv converges to the inverse $\mathbf{A}^{-1}$ or $\mathbf{A}^{\dagger}$ extremely fast.

**Lemma 1** (Approximation Speed). *The generated sequence $\{\mathbf{X}_k \in \mathbb{R}^{d_1\times d_2}\}$ by Eq. (2) converges to the Moore–Penrose inverse $\mathbf{A}^{\dagger}$ with order $L$ provided that $\mathbf{X}_0 = \frac{1}{c_0}\mathbf{A}^{\top}$, $c_0 > \frac{1}{2}\|\mathbf{A}\|^2$, i.e., we can conclude:*

$$\left\|\mathbf{AA}^{\dagger} - \mathbf{AX}_k\right\| = e_0^{L^k}, \quad \textit{in which} \quad e_0 = \left\|\mathbf{AA}^{\dagger} - \mathbf{AX}_0\right\| < 1,$$

*where $\|\cdot\|$ is the spectral norm.*

We can see that fixed D-Minv converges with order $L$, i.e., a shallow D-Minv can approximate the matrix inverse very well. Moreover, provided with the $\mathbf{X}_0$ given in Lemma 1, the column space and the row space of the matrix $\mathbf{X}_k$ are exactly correct from the beginning.

**Proposition 1** (Invariant Column and Row Spaces). *With the same setting in Lemma 1, for any $k \geq 0$, it holds that:*

$$\mathbf{X}_k\mathbf{AA}^{\dagger} = \mathbf{X}_k, \quad \mathbf{A}^{\dagger}\mathbf{AX}_k = \mathbf{X}_k.$$

By Proposition 1, it is easy to see that the zero singular values remain unchanged during computing, which indicates that D-Minv works consistently on the full and rank-deficient square and rectangle matrices. Note that this proposition also holds for the upcoming LD-Minv.

### 2.2 Learnable Differentiable Matrix Inverse

We consider the learnable D-Minv (LD-Minv) with the following iterative formula:

$$\mathbf{X}_{k+1} = \mathbf{X}_k\left(\sum_{l=0}^{L-1}C_{\{k,l\}}\mathbf{E}_k^l\right), \quad \mathbf{E}_k = \mathbf{I} - \mathbf{AX}_k, \quad L \geq 4, \tag{3}$$

[1]Higher-order method Eq. (2) is already well-known in the applied mathematics community in another equivalent form, see Climent et al. (2001); Amat et al. (2003); Li et al. (2011).

where $(k, l+1) \in [K] \times [L]$, $C_{\{k,l\}} \in \mathbb{R}$ are learnable coefficients, and $\|\cdot\|_F$ is the Frobenius norm.

LD-Minv is also in the category of LbMs that unroll the conventional numerical iteration methods. As discussed in the introduction, two problems, i.e., no convergence analysis for the training procedure and no generalization guarantee for the trained DNN, block the further theoretical exploration of these LbMs. However, in contrast to the previous works, we obtain favorable theoretical results on both of the two problems for our LD-Minv. First, although fixed D-Minv already converges extremely fast, LD-Minv can still perform much better on the training data under mild conditions. Specifically, $\left\|\mathbf{X}_K - \mathbf{A}^\dagger\right\|$ converges to zero exponentially fast during training, i.e., GD finds an $\epsilon$-error minimum in $\ell_2$ regression using at most $\mathcal{O}(nKL \log 1/\epsilon)$ iterations [2]. Second, LD-Minv has a tight generalization bound to guarantee the performance on the unseen matrix when it comes from the same distribution as the training instances. We provide the rigorous theoretical results in Sec. 4.

**Discussion 1.** One can find that the final iterate $\mathbf{X}_K$ can be written as matrix polynomial of $\mathbf{A}$, and may argue that approximating the Minv by the polynomial is well-studied. Zero training can be obtained when the polynomial's order is large enough, and the approximation error controls the generalization error. For the sake of distinction, we make three clarifications. First of all, most traditional results only describe the behavior in the **worst** case, which holds for *any* input matrix and may easily fail in some extreme cases. However, LD-Minv focuses on a more realistic situation, i.e., the input matrices obey some common distribution, such as the affinity matrices of Facebook users or the sampled CT and MRI images. How to learn from the data and design a more efficient method is still open, not to mention the theoretical generalization guarantee on specific data distribution. Secondly, the traditional polynomial approximation method is susceptible to the polynomial order and is very easy to over-fitting when the order is over-parameterized, say order$\gg nd$. Without precise data distribution density, it is impossible to choose the proper order to ensure zero training and guarantee the small generalization error, simultaneously. To solve this, we may need some complex and data-driven regularization strategy. In sharp contrast, LD-Minv is robust to the order, and our theoretical results (i.e., zero training and small generalization error) hold well from under-parameterized to over-parameterized cases. Thirdly, the exact polynomial fitting can only imply the existence of zero training solution, while our results describe the convergence behavior (i.e., which solution we will choose) . Obviously, there exists a big gap between the convergence of training and the existence of a solution.

Note that we have considered a much easier matrix polynomial: $\mathbf{X}_K = \sum_{l=0}^{L} C_l \mathbf{A}^l$. Learning the coefficients $\{C_l\}_{l=1}^{L}$ becomes an easy-to-solve convex regression problem unlike the convex one in Eq. (3). Unfortunately, we failed due to some stability issues, e.g, the coefficients in different powers vary significantly which cause the poor generalization performance.

**Discussion 2.** Most matrix iterative methods suffer from the ill-condition problem, which usually brings instability and makes the analysis and calculation hard to carry on, e.g., for D-Minv, large condition number may significantly slow the convergence speed (see Lemma 1). However, learning-based method can greatly alleviate this ill-condition problem by learning from data. LD-Minv can beat D-Minv easily when the condition number of the input matrix is large and the learned coefficients for LD-Minv are also valid on the unseen data. See the experimental part for more details.

## 2.3 Training Settings

We now describe some training settings for D-Minv. Denote by $\{\mathbf{A}_i\}_{i=1}^{n}$ the training set consisting of $n$ samples. Let $\mathbf{X}_{\{k,i\}} \in \mathbb{R}^{d_1 \times d_2}$ be the output of the $k$-th layer on the $i$-th training sample $\mathbf{A}_i \in \mathbb{R}^{d_2 \times d_1}$. We consider a typical regression loss:

$$\mathcal{L}_n(\mathbf{C}) := \frac{1}{2nK} \sum_{i=1}^{n} \sum_{k=1}^{K} \left\|\mathbf{A}_i \mathbf{X}_{\{k,i\}} \mathbf{A}_i - \mathbf{A}_i\right\|_F^2, \tag{4}$$

where $\mathbf{C} = \{C_{\{k,l\}},\ (k, l+1) \in [K] \times [L]\}$ is the collection of all learnable parameters. Note that when $\mathbf{X} = \mathbf{A}^\dagger$, the properties of the Moore–Penrose inverse indicate the equation $\mathbf{AXA} = \mathbf{A}$, which can further indicates the equation $\mathbf{XAX} = \mathbf{X}$ in our invariant space case, see Proposition 1. Therefore, we only use the difference term $(\mathbf{AXA} - \mathbf{A})$ in the training loss.

[2]Please distinguish the two "convergence" here. One describes $\mathbf{X}_k$ approaching $\mathbf{A}^\dagger$ w.r.t. $k$ and $L$, and the other refers to the behavior of LD-Minv's loss w.r.t. training iteration.

**Algorithm 1** Gradient descent (GD) with proper initialization for LD-Minv

```
Input: Training data $\{\mathbf{A}_i\}_{i=1}^n$, number of iterations $T$, step size $\eta$.
1: Generate each $\mathbf{C}_{\{k,l\}} \in \mathbf{C}$ such that $|\mathbf{C}_{\{k,l\}} - 1| = \mathcal{O}(K^{-\alpha})$, where $\alpha > \frac{1}{4}$.
2: Set $\mathbf{X}_{\{0,i\}} = \frac{1}{c_0}\mathbf{A}_i^\top$ or $\widetilde{\mathbf{X}}_i$, where $c_0 > \frac{1}{2}\|\mathbf{A}_i\|^2$, $\widetilde{\mathbf{X}}_i$ is the output of a fixed D-Minv.
3: for $t = 1, \cdots, T$ do
4:     Update $\mathbf{C}^{(t+1)} = \mathbf{C}^{(t)} - \eta \cdot \mathcal{D}_{[\mathbf{C}=\mathbf{C}^{(t)}]}\mathcal{L}_n$.
5: end for
Output: $\mathbf{C}^{(0)}, \dots, \mathbf{C}^{(T)}$.
```

Thanks to the differentiability of LD-Minv, we adopt GD to minimize the training loss to obtain the proper coefficients $\mathbf{C}$. We present the initialization strategy and training process in Algorithm 1, where the first derivative of differentiable function $\mathcal{L}_n(\mathbf{C}) : \mathbb{R}^{K\times L} \to \mathbb{R}$ at $\mathbf{C}^{(t)}$ is denoted by $\mathcal{D}_{[\mathbf{C}=\mathbf{C}^{(t)}]}\mathcal{L}_n(\cdot) := \big(\partial\mathcal{L}_n(\mathbf{C}^{(t)})/\partial\mathbf{C}\big) \in \mathbb{R}^{K\times L}$. When $K$ is large, one may notice that our coefficient is close to the oracle value 1, which may provide a relatively good initial loss $\mathcal{L}\big(\mathbf{C}^{(0)}\big)$. Notably, this does **NOT** mean that we will treat the learning and initialization as a perturbation of the fixed D-Minv and bound the loss by the good-enough fixed D-Minv's performance plus the perturbation error. On the contrary, as we will show in theoretical results part, learnability indeed matters, and LD-Minv can obtain better performance than the fixed one by training. The real purpose of this initialization is to take advantage of D-Minv's local Lipschitz continuity near $\mathbf{C} = \mathbf{1}$, where $\mathbf{1}$ is the all one vector. The continuity reduces the complexity of optimization and will benefit the generalization of D-Minv.

## 3 Learning-based Optimization with Orthogonality Constraints

Considering the following general orthogonality constrained optimization problem:

$$\min_{\mathbf{U}} f(\mathbf{U}), \quad \text{s.t. } \mathbf{U}^\top\mathbf{U} = \mathbf{I}, \tag{5}$$

where the objective function $f(\mathbf{U}) : \mathbb{R}^{m\times r} \to \mathbb{R}$ is differentiable and convex. The usual way to solve Eq. (5) is to perform the manifold GD on the Stiefel manifold, which evolves along the manifold geodesics. Specifically, manifold GD first updates the variable in the manifold tangent space along the objective function's projected gradient. Then, map the updated variable in the tangent space to a feasible point on the geodesic, and repeat these two steps until convergence (Edelman et al., 1998). Usually, the mapping step is non-differentiable and inefficient. Fortunately, the work (Wen & Yin, 2013; Ren & Lin, 2013) develops a technique to solve the optimization problem with orthogonal constrains approximately, which only involves matrix multiplication and inverse.

We let $\mathbf{U} \in \mathbb{R}^{m\times r}$, where $r$ is the Stiefel manifold's dimension. Denote by $\mathbf{G} \in \mathbb{R}^{m\times r}$ the gradient of the objective function $f(\mathbf{U})$ in Eq. (5) w.r.t. $\mathbf{U}$ at $\mathbf{U}_k$, then the projection of $\mathbf{G}$ in the tangent space of the Stiefel manifold at $\mathbf{U}_k$ is $\mathbf{P}\mathbf{U}_k$[3], where $\mathbf{P} = \mathbf{G}\mathbf{U}_k^\top - \mathbf{U}_k\mathbf{G}^\top$ and $\mathbf{P} \in \mathbb{R}^{m\times m}$. Instead of parameterizing the geodesic of the Stiefel manifold along direction $\mathbf{P}$ using the exponential function, we generate the feasible points to update $\mathbf{U}_k$ by the following Cayley transform (Wen & Yin, 2013):

$$\mathbf{U}_{k+1} = \mathbf{U}(t) = \mathbf{C}(t)\mathbf{U}_k, \quad \text{where} \quad \mathbf{C}(t) = \left(\mathbf{I} + \frac{t}{2}\mathbf{P}\right)^{-1}\left(\mathbf{I} - \frac{t}{2}\mathbf{P}\right), \tag{6}$$

where $\mathbf{I}$ is the identity matrix and $t \in \mathbb{R}$ is the step size used for updating the current $\mathbf{U}_k$. In other words, $\mathbf{U}(t)$ is a re-parameterized local geodesic w.r.t. $t$ on the Stiefel manifold. One can easily verify that $\mathbf{U}(t)$ has the following properties, given $\mathbf{U}_k^\top\mathbf{U}_k = \mathbf{I}$:
(1) $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{U}(0) = -\mathbf{P}\mathbf{U}_k$, (2) $\mathbf{U}(t)$ is smooth in $t$, (3) $\mathbf{U}(0) = \mathbf{U}_k$, (4) $\mathbf{U}(t)^\top\mathbf{U}(t) = \mathbf{I}$, $\forall t \in \mathbb{R}$.
It is evident that, if $t$ is in a proper range, $\mathbf{U}_{k+1}$ can lead to a lower objective function value than $\mathbf{U}(0) = \mathbf{U}_k$ on the Stiefel manifold. Besides, computing $\mathbf{U}_{k+1}$ only involves the matrix multiplication and matrix inverse, which can be easily performed by our LD-Minv in Eq. (3. Therefore, we let $\mathbf{C}(t) = \text{LD-Minv}\,(\mathbf{I} + t\mathbf{P}/2)$ in Eq. (6). Then we can obtain a learning-based optimization method for the problem with orthogonality constraints in Eq. (5).

[3] We choose the canonical metric on the tangent space as the equipped Riemannian metric.

## 3.1 Application: Differentiable SVD by LD-Minv

In this part, we show how to utilize the previous general learning-based optimization framework to differentiate SVD. Given an arbitrary matrix $\mathbf{M} \in \mathbb{R}^{m \times \hat{n}}$ and its **skinny** SVD $\mathbf{M} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^\top$, the Von Neumann's trace inequality, $\langle \mathbf{M}, \mathbf{X} \rangle \le \sum_i \sigma_i(\mathbf{M})\sigma_i(\mathbf{X})$, implies that $\{\tilde{\mathbf{U}} \in \mathbb{R}^{m \times r}, \tilde{\mathbf{V}} \in \mathbb{R}^{\hat{n} \times r}\}$ is an optimal solution of the following optimization problem:

$$\min_{\mathbf{U},\mathbf{V}} f(\mathbf{U},\mathbf{V}) \coloneqq \left(\frac{1}{2}\|\mathrm{D^c}(\mathbf{\Lambda})\|_F^2 - \langle \mathbf{M}, \mathbf{U}\mathbf{V}^\top \rangle\right), \quad \text{s.t. } \mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}, \tag{7}$$

where $\mathbf{\Lambda} \coloneqq \mathbf{U}^\top\mathbf{M}\mathbf{V}$, $\mathrm{Diag}(\cdot)$ returns the collection of the diagonal entries of the input matrix, and $\mathrm{D^c}(\mathbf{\Lambda}) \coloneqq (\mathrm{Id} - \mathrm{Diag})(\mathbf{\Lambda}) \coloneqq \mathbf{\Lambda} - \mathrm{Diag}(\mathbf{\Lambda})$ for any input matrix $\mathbf{\Lambda}$.

Note that solving the problem in Eq. (7) is equivalent to performing SVD. Let $\{\mathbf{M}, \mathbf{U}_0, \mathbf{V}_0\}$ be the inputs of our Differentiable SVD (D-SVD). We adopt the Cayley transform in Eq. (6) to solve the problem in Eq. (7), and replace the matrix inverse in it by our proposed LD-Minv in Eq. (3). W.l.o.g, in the following, we only focus on the updating strategy of $\mathbf{U}$ due to the similarity between $\mathbf{U}$ and $\mathbf{V}$. We first compute the gradient of the objective function $f(\mathbf{U},\mathbf{V})$ w.r.t. $\mathbf{U}$ at $\{\mathbf{U}_k, \mathbf{V}_k\}$, which we shall denote by $\mathbf{G} = \mathbf{M}\mathbf{V}_k\big(\mathrm{D^c}\left(\mathbf{V}_k^\top\mathbf{M}^\top\mathbf{U}_k\right) - \mathbf{I}\big)$. Then we find a geodesic curve $\mathbf{U}(t)$ along the gradient on the Stiefel manifold for updating $\mathbf{U}_k$. Referring to Eq. (6), the curve is $\mathbf{U}(t) = \big(\mathbf{I} + \frac{t}{2}\mathbf{P}\big)^{-1}\big(\mathbf{I} - \frac{t}{2}\mathbf{P}\big)\mathbf{U}_k$, where $\mathbf{P} = \mathbf{G}\mathbf{U}_k^\top - \mathbf{U}_k\mathbf{G}^\top$. Given the step size $t$, we can use LD-Minv to perform the matrix inverse on the factor $\big(\mathbf{I} + \frac{t}{2}\mathbf{P}\big)$. In summery, D-SVD consists of two steps: (1) find a proper $t$; (2) update $\{\mathbf{U}_k, \mathbf{V}_k\}$ by the Cayley transform.

For finding a proper step size $t$, we consider the following problem:

$$t_U^* = \operatorname*{argmin}_{0 \le t \le \varepsilon} f(t) \coloneqq f(\mathbf{U}(t), \mathbf{V}_k), \tag{8}$$

where $\varepsilon$ is a given parameter to ensure a small enough magnitude of $t_U^*$. Notice that if $t$ is small enough to hold $\left\|\frac{t}{2}\mathbf{P}\right\| < 1$, then we have $\big(\mathbf{I} + \frac{t}{2}\mathbf{P}\big)^{-1} = \mathbf{I} + \sum_{l=1}^{\infty}\left(-\frac{t}{2}\mathbf{P}\right)^l$, which implies that $\mathbf{U}(t) = \Big(\mathbf{I} + 2\sum_{l=1}^{\infty}\left(-\frac{t}{2}\mathbf{P}\right)^l\Big)\mathbf{U}_k$. Considering that $t^*$ in Eq. (8) is small, we can approximate $f(t)$ via its second order Taylor expansion at $t = 0$:

$$f(t) = f(0) + f'(0) \cdot t + \frac{1}{2}f''(0) \cdot t^2 + \mathcal{O}\big(t^2\big), \tag{9}$$

where $f'(0)$ and $f''(0)$ are the first and the second order derivatives of $f(t)$ evaluated at 0, respectively. These two derivatives have closed form and can be computed efficiently. Consequently, we can obtain an approximated optimal solution $t^*$ via:

$$t_U^* = \min\left\{\varepsilon, \tilde{t}\right\}, \quad \text{where} \quad \varepsilon < 2/\|\mathbf{P}\|, \quad \text{and} \quad \tilde{t} = -f'(0)/f''(0), \tag{10}$$

provided that:

$$\begin{cases} f'(0) = \left\langle \mathrm{D^c}\left(\mathbf{U}_k^\top\mathbf{M}\mathbf{V}_k\right), \mathrm{D^c}\left(\mathbf{U}_k^\top\mathbf{P}\mathbf{M}\mathbf{V}_k\right)\right\rangle + \langle \mathbf{M}\mathbf{V}_k, \mathbf{P}\mathbf{U}_k \rangle, \\ f''(0) = \left\|\mathrm{D^c}\left(\mathbf{U}_k^\top\mathbf{P}\mathbf{M}\mathbf{V}_k\right)\right\|_F^2 + \left\langle \mathrm{D^c}\left(\mathbf{U}_k^\top\mathbf{M}\mathbf{V}_k\right), \mathrm{D^c}\left(\mathbf{U}_k^\top\mathbf{P}^2\mathbf{M}\mathbf{V}_k\right)\right\rangle - \left\langle \mathbf{M}\mathbf{V}_k, \mathbf{P}^2\mathbf{U}_k \right\rangle. \end{cases} \tag{11}$$

Then we can update $\mathbf{U}$ by $\mathbf{U}(t_U^*)$, i.e., $\mathbf{U}_{k+1} = \mathbf{U}(t_U^*)$. Thanks to the cyclic property of trace operator, $\mathbf{V}_{k+1}$ shares a similar update strategy with $\mathbf{U}_{k+1}$.

Provided the input $\{\mathbf{M}, \mathbf{U}_k, \mathbf{V}_k\}$, the complete iterative steps for our D-SVD are as follows[4]:

$$\begin{cases} \mathbf{G}_U = \mathbf{M}\mathbf{V}_k\big(\mathrm{D^c}\left(\mathbf{V}_k^\top\mathbf{M}^\top\mathbf{U}_k\right) - \mathbf{I}\big), & \mathbf{P}_U = \mathbf{G}_U\mathbf{U}_k^\top - \mathbf{U}_k\mathbf{G}_U^\top, \\ t_U^* = \min\left\{\dfrac{2}{\|\mathbf{P}_U\|}, -\dfrac{f'(\mathbf{U}(t), \mathbf{V}_k)}{f''(\mathbf{U}(t), \mathbf{V}_k)}\right\} + \tilde{t}_{U_k}, & \mathbf{U}_{k+1} = \mathrm{Cayley}\left(t_U^*, \mathbf{C}_{U_k}, \mathbf{P}_U, \mathbf{U}_k\right), \\ \mathbf{G}_V = \mathbf{M}^\top\mathbf{U}_{k+1}\big(\mathrm{D^c}\left(\mathbf{U}_{k+1}^\top\mathbf{M}\mathbf{V}_k\right) - \mathbf{I}\big), & \mathbf{P}_V = \mathbf{G}_V\mathbf{V}_k^\top - \mathbf{V}_k\mathbf{G}_V^\top, \\ t_V^* = \min\left\{\dfrac{2}{\|\mathbf{P}_V\|}, -\dfrac{f'(\mathbf{U}_{k+1}, \mathbf{V}(t))}{f''(\mathbf{U}_{k+1}, \mathbf{V}(t))}\right\} + \tilde{t}_{V_k}, & \mathbf{V}_{k+1} = \mathrm{Cayley}\left(t_V^*, \mathbf{C}_{V_k}, \mathbf{P}_V, \mathbf{V}_k\right), \end{cases} \tag{12}$$

[4]For convenience and clearing writing, we omit the superscript in the updating rules.

**Algorithm 2** Forward Propagation for D-SVD

```
Input: Training data $\{\mathbf{M}_i, \mathbf{U}_{i,0}, \mathbf{V}_{i,0}\}_{i=1}^n$, depth $K_{\rm svd}$ of D-SVD, number of iterations $T_{\rm inv}$, step size $\eta_{\rm inv}$ and D-SVD's current parameters $\tilde{\mathbf{t}}$.
1: for $k = 0, \ldots, K_{\rm svd}$ do
2:     Calculate $\mathbf{P}_{U,i}$ and $t^*_{U,i}$ with $\{\mathbf{M}_i, \mathbf{U}_{i,k}, \mathbf{V}_{i,k}\}_{i=1}^n$ by Eq. (12).
3:     Train LD-Minv with parameters $\mathbf{C}_{U_k}$ by Algorithm 1 with the data $\{\mathbf{A}_i\}_{i=1}^n$, number of iterations $T_{\rm inv}$ and step size $\eta_{\rm inv}$, where $\mathbf{A}_i = (\mathbf{I} + t^*_{U,i}\mathbf{P}_{U,i}/2)$.
4:     Repeat Steps 2 and 3 by switching the roles of $\mathbf{U}$ and $\mathbf{V}$ with the input $\{\mathbf{U}_{i,k+1}, \mathbf{V}_{i,k}\}_{i=1}^n$.
5: end for
Output: $\{\mathbf{C}_{U_k}, \mathbf{C}_{V_k}\}_{k=1}^{K_{\rm svd}}$ and $\{\mathbf{U}_{i,k+1}, \mathbf{V}_{i,k}\}_{i=1,k=1}^{n,K_{\rm svd}}$.
```

**Algorithm 3** Joint Training for D-SVD and LD-Minv

```
Input: Training data $\{\mathbf{M}_i\}_{i=1}^n$, number of iterations $T_{\rm inv}$ and $T_{\rm svd}$, step sizes $\eta_{\rm svd}$ and $\eta_{\rm inv}$.
1: Generate $\{\mathbf{U}_{i,0}, \mathbf{V}_{i,0}\}_{i=1}^n$ randomly from the $r$-dimensional Stiefel manifold. Set $\tilde{\mathbf{t}}^{(0)} = \mathbf{0}$.
2: for $t = 1, \cdots, T_{\rm svd}$ do
3:     Forward propagate by Algorithm 2.
4:     Update $\tilde{\mathbf{t}}^{(t+1)} = \tilde{\mathbf{t}}^{(t)} - \eta_{\rm svd} \cdot \mathcal{D}_{[\tilde{\mathbf{t}}=\tilde{\mathbf{t}}^{(t)}]}\mathcal{M}_n$, where $\mathcal{M}_n := \frac{1}{nK_{\rm svd}} \sum_{i=1}^n \sum_{k=1}^{K_{\rm svd}} f(\mathbf{U}_{i,k}\mathbf{V}_{i,k})$.
5: end for
Output: $\{\mathbf{U}_{i,k+1}, \mathbf{V}_{i,k}\}_{i=1,k=1}^{n,K_{\rm svd}}$.
```

where

$$\text{Cayley}\,(t, \mathbf{C}, \mathbf{P}, \mathbf{U}) := \text{LD-Minv}\,(\mathbf{I} + t\mathbf{P}/2, \mathbf{C})(\mathbf{I} - t\mathbf{P}/2)\mathbf{U},$$

$\tilde{\mathbf{t}} := \{(\tilde{t}_{U_k}, \tilde{t}_{V_k}) \in \mathbb{R}^2\}_{k=1}^{K_{\rm svd}}$ is the collection of learnable parameters for D-SVD, and $\text{LD-Minv}(\cdot, \mathbf{C})$ is a LD-Minv module with parameters $\mathbf{C}$. Note that a DNN can easily implement our D-SVD. Each layer implements the steps in Eq. (12), which are the specific iterative procedures of the previous general learning-based optimization with orthogonality constraints. We adopt different LD-Minvs for different layers, and provide the training process for D-SVD in Algorithms 2 and 3. By introducing LD-Minv into the Cayley transform, we bypass the non-differentiable exact Minv, and solve the problem in Eq. (7) (i.e., perform SVD) in a differentiable and learning-based way.

## 4 Main Results

In this section, we provide the main theoretical results of D-Minv, including the linear convergence of GD during training and the generalization performance. All the detailed proofs of these theorems are provided in the supplementary material. The theoretical results for D-SVD are presented in the supplementary material due to limited space.

### 4.1 Convergence Rate of GD

We consider the general LD-Minv and large training data size $n$ in this section. We first make the following assumption on the training data.

**Assumption 1** (Bounded Singular Values). *Given the training matrices $\left\{\mathbf{A}_i \in \mathbb{R}^{d_1 \times d_2}\right\}_{i=1}^n$, we assume the training matrices'* ***positive*** *singular value vectors are d-dimensional, where $d \leq \min\{d_1, d_2\}$. Assume these positive singular values have lower and upper bounds, i.e.,*

$$\|\mathbf{x}_i\|_\infty \leq 1 \quad \textit{and} \quad \min_{j \in [d]} [\mathbf{x}_i]_j \geq \bar{b}_a > 0, \quad \textit{where} \quad \mathbf{x}_i = \sigma_+(\mathbf{A}_i) \in \mathbb{R}^d, \quad \forall i \in [n],$$

*and $\sigma_+(\cdot)$ extract the positive singular values.*

In general, the boundedness assumption for the singular values the is weak and easy to satisfy. See the discussion in the supplementary material (Section C.1.2).

With this assumption, we can obtain the convergence rate of our Algorithm 1.

**Theorem 1** (Convergence Rate)**.** *Suppose Assumption 1 holds and let* $d = \min\{d_1, d_2\}$*. For any* $t, \epsilon > 0$*, we let* $K = \Omega\big(d\bar{b}_a^2\big)$*, then with probability at least* $1 - \exp\big(-\mathcal{O}\big(K/\bar{b}_a^2\big)\big)$*, GD in Algorithm 1 with step size* $\eta = \Theta(1/dL)$ *can find a collection of coefficients such that:*

$$\mathcal{L}_n(\mathbf{C}^{(T)}) < \epsilon, \quad for \quad T = \Theta\big(dLKn\bar{b}_a^{-2}\ln(1/\epsilon)\big),$$

*where* $\bar{b}_a$ *is the lower bound for the positive singular values of training data* $\mathbf{A}_i$*.*

This is known as the linear convergence rate because $\epsilon$ drops exponentially fast in $T$. Recall that $K$ and $L$ are from the definition of our LD-Minv.

**Remark 1.** *Different from the traditional theoretical results of DNNs, the randomness in our results mainly comes from a re-weighting strategy (see Sec. E) rather than initialization. In general, our results hold for any initialization strategy as long as the condition in Algorithm 1 is met. Notably, our results do not require the depth or the width to be in the polynomial order of data size* $n$*, which is a prevalent over-parameterization assumption for the current deep learning theories.*

**Remark 2.** *To present our convergence result most simply, we choose to focus on GD method with fixed step size mainly. It shall be easy to extend our results to more general settings, such as stochastic GD and dynamic step size.*

### 4.2 Generalization

We characterize the generalization performance of LD-Minv trained by GD in this theorem.

**Theorem 2** (Generalization Bound)**.** *Denote by* $\mathcal{P}_{\mathbf{A}}$ *the distribution of* $\mathbf{A}_i$ *in Assumption 1. Suppose that* $\alpha \geq 1$ *and* $K^{2\alpha-1/2} = \Omega(\sqrt{n})$*, then with probability at least* $1-\delta$*, the iterate* $\mathbf{C}^{(t)}$ *of Algorithm 1 holds that:*

$$\mathop{\mathbb{E}}_{\mathbf{A}\sim\mathcal{P}_{\mathbf{A}}}\left[\mathcal{L}_n\Big(\mathbf{C}^{(t)}\Big)\right] \leq \mathcal{L}_n\Big(\mathbf{C}^{(t)}\Big) + \widetilde{\mathcal{O}}\left(\min\left\{\frac{\bar{b}_w^2 d^3 L}{K^{\alpha/2}}, \sqrt{\frac{d^3 K}{n}}\right\}\right) + \mathcal{O}\left(\sqrt{\frac{\ln(1/\delta)}{n}}\right),$$

*for* $t = 0, 1, \cdots, T$*, where* $\mathbb{E}_{\mathbf{A}\sim\mathcal{P}_{\mathbf{A}}}[\mathcal{L}_n]$ *is the expected value of* $\mathcal{L}_n$ *under the probability measure* $\mathcal{P}_{\mathbf{A}}$ *and* $\widetilde{\mathcal{O}}(\cdot)$ *hides log factors.*

We adopt two ways to upper bound the empirical Rademacher complexity, which corresponds to the cases $K < n$ and $K > n$, respectively. The final bound is the minimum of them (middle term in the above upper bound).

**Remark 3.** *The second term in our bound distinguishes our result from most previous work (Allen-Zhu et al., 2019; Arora et al., 2019; Yehudai & Shamir, 2019; Cao & Gu, 2019; Chen et al., 2019; Xie et al., 2020) on the generalization bounds of over-parameterized neural networks. Specifically, most over-parameterized work mainly focus on establishing the bound that does not explode when the network width goes to infinity. However, their bounds are exponential in the depth, which may not hold when the depth is large. In contrast, our result covers a wider range of depth* $K$*: it covers the case of both* $K < n$ *and* $K > n$*. One may wonder whether our bound will explore when* $K$ *approaches infinity. The answer is no. We can observe a "double descent" trend to certain extent: the generalization error bound first increases with the network depth* $K$ *when* $K < n$*, then it starts to decrease when* $K$ *becomes larger even for* $K \gg n$*.*

## 5 Conclusion

We provide a differentiable and learnable way, named LD-Minv, to perform the matrix inverse by using a $L$-th order matrix polynomials. Then, we offer some theoretical guarantees about the convergence during training and generalization ability in both under-parameterization and over-parameterization setting for LD-Minv. Remarkably, we are the first to provide the strict analysis for the training process of the differentiable learning-based methods. As an application of LD-Minv, we propose a learning-based optimization method to solve the problems with non-convex orthogonality constraints, and then utilize it to differentiate SVD. A generalization guarantee for D-SVD is also provided.

## REFERENCES

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pp. 6155–6166, 2019.

Sergio Amat, Sonia Busquier, and JM Gutiérrez. Geometric constructions of iterative functions to solve nonlinear equations. *Journal of Computational and Applied Mathematics*, 157(1):197–205, 2003.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pp. 322–332, 2019.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 10835–10845, 2019.

Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In *Advances in Neural Information Processing Systems*, pp. 9061–9071, 2018.

Zixiang Chen, Yuan Cao, Difan Zou, and Quanquan Gu. How much over-parameterization is sufficient to learn deep ReLU networks? *arXiv preprint arXiv:1911.12360*, 2019.

Joan-Josep Climent, Néstor Thome, and Yimin Wei. A geometrical approach on generalized inverses by neumann-type series. *Linear Algebra and its Applications*, 332:533–540, 2001.

Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 163–172, 2015.

R Brent Dozier and Jack W Silverstein. Analysis of the limiting spectral distribution of large dimensional information-plus-noise type matrices. *Journal of Multivariate Analysis*, 98(6):1099–1122, 2007.

Richard M Dudley. The sizes of compact subsets of hilbert space and continuity of gaussian processes. *Journal of Functional Analysis*, 1(3):290–330, 1967.

Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Gene H Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the International Conference on Machine Learning*, pp. 399–406, 2010.

Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. In *Advances in Neural Information Processing Systems*, pp. 7400–7410, 2019.

Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2965–2973, 2015.

Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *arXiv preprint arXiv:1909.12292*, 2019.

Hou-Biao Li, Ting-Zhu Huang, Yong Zhang, Xing-Ping Liu, and Tong-Xiang Gu. Chebyshev-type methods and preconditioning techniques. *Applied Mathematics and Computation*, 218(2):260–270, 2011.

Jun Li, Fuxin Li, and Sinisa Todorovic. Efficient riemannian optimization on the stiefel manifold via the cayley transform. In *International Conference on Learning Representations*, 2019.

Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 612–620, 2011.

Guangcan Liu, Zhouchen Lin, and Yong Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of the International Conference on Machine Learning*, pp. 663–670, 2010.

Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013.

Jialin Liu, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Alista: Analytic weights are as good as learned weights in lista. In *International Conference on Learning Representations*, 2019.

Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 91–100, 2013.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. 2018.

Jelani Nelson and Huy L Nguyen. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 ieee 54th annual symposium on foundations of computer science*, pp. 117–126. IEEE, 2013.

Xi Peng, Joey Tianyi Zhou, and Hongyuan Zhu. K-meansNet: When K-means meets differentiable programming. *arXiv preprint arXiv:1808.07292*, 2018.

Xiang Ren and Zhouchen Lin. Linearized alternating direction method with adaptive penalty and warm starts for fast solving transform invariant low-rank textures. *International Journal of Computer Vision*, 104(1):1–14, 2013.

Mark Rudelson and Roman Vershynin. The Littlewood–Offord problem and invertibility of random matrices. *Advances in Mathematics*, 218(2):600–633, 2008.

Uwe Schmidt and Stefan Roth. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2774–2781, 2014.

Jianhong Shen. On the singular values of gaussian random matrices. *Linear Algebra and its Applications*, 326(1-3):1–14, 2001.

GW Stewart. On the continuity of the generalized inverse. *SIAM Journal on Applied Mathematics*, 17(1):33–45, 1969.

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Zhangyang Wang, Ding Liu, Shiyu Chang, Qing Ling, Yingzhen Yang, and Thomas S Huang. D3: Deep dual-domain based fast restoration of jpeg-compressed images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2764–2772, 2016.

Colin Wei and Tengyu Ma. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. In *Advances in Neural Information Processing Systems*, pp. 9722–9733, 2019.

Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

Jianlong Wu, Xingyu Xie, Liqiang Nie, Zhouchen Lin, and Hongbin Zha. Unified graph and low-rank tensor learning for multi-view clustering. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pp. 6388–6395, 2020a.

Kailun Wu, Yiwen Guo, Ziang Li, and Changshui Zhang. Sparse coding with gated learned ISTA. In *International Conference on Learning Representations*, 2020b.

Xingyu Xie, Jianlong Wu, Guangcan Liu, Zhisheng Zhong, and Zhouchen Lin. Differentiable linearized ADMM. In *International Conference on Machine Learning*, pp. 6902–6911, 2019.

Xingyu Xie, Hao Kong, Jianlong Wu, Guangcan Liu, and Zhouchen Lin. Maximum-and-concatenation networks. In *International Conference on Machine Learning*, pp. 1358–1368, 2020.

Yan Yang, Jian Sun, Huibin Li, and Zongben Xu. Deep ADMM-Net for compressive sensing MRI. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 10–18, 2016.

Gilad Yehudai and Ohad Shamir. On the power and limitations of random features for understanding neural networks. In *Advances in Neural Information Processing Systems*, pp. 6594–6604, 2019.

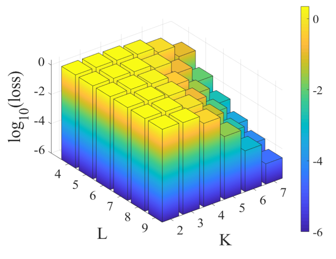
Xiao Zhang, Lingxiao Wang, Yaodong Yu, and Quanquan Gu. A primal-dual analysis of global optimality in nonconvex low-rank matrix recovery. In *Proceedings of the International Conference on Machine Learning*, pp. 5857–5866, 2018.

## Appendix

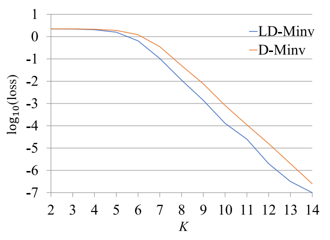
## A Experimental Results

### A.1 Effectiveness Validation

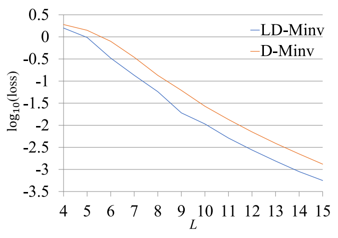
We conduct experiments on synthetic data to validate the effectiveness of our proposed LD-Minv and D-SVD. We run our all experiments for 10 times and report the average results.



(a)



(b)



(c)

Figure 1: Experimental results of LD-Minv and D-Minv. (a) Test loss of LD-Minv with different $K$ and $L$; (b) Comparision between LD-Minv and D-Minv with different $K$ and fixed $L = 4$; (c) Comparison between LD-Minv and D-Minv with different $L$ and fixed $K = 5$.

#### A.1.1 Experiments Results for LD-Minv

**Settings**. For LD-Minv, we adopt the square and rectangle matrix to test its performance. It should be mentioned that our LD-Minv is adaptive to the shape of the input matrix. In general, the square matrix can better test the performance of the algorithm, since the minimum singular value is near 0 in this case (Vershynin, 2010). And a large condition number of the input may make most numerical matrix calculation methods unstable. For the square matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, its elements are sampled from i.i.d. Gaussian, namely $A_{i,j} \sim \mathcal{N}(0, 1/\sqrt{d})$. For the rectangle matrix $\mathbf{A} \in \mathbb{R}^{m\times \hat{n}}$, its elements

Table 1: Inference time (seconds) of D-Minv/LD-Minv and exact Minv for different matrix size with batch size 100. The efficiency advantage of our LD-Minv/D-Minv is obvious through all cases.

| Matrix Size | $10 \times 15$ | $50 \times 75$ | $100 \times 150$ | $200 \times 300$ | $500 \times 750$ | $1000 \times 1500$ |
|---|---|---|---|---|---|---|
| $L = 4, K = 4$ | 0.0030 | 0.0070 | 0.010 | 0.025 | 0.40 | 1.23 |
| $L = 4, K = 8$ | 0.0052 | 0.0088 | 0.016 | 0.047 | 0.56 | 2.22 |
| $L = 4, K = 12$ | 0.0083 | 0.0133 | 0.019 | 0.055 | 0.64 | 3.08 |
| $L = 4, K = 5$ | 0.0035 | 0.0050 | 0.013 | 0.025 | 0.38 | 1.38 |
| $L = 8, K = 5$ | 0.0058 | 0.0076 | 0.014 | 0.051 | 0.55 | 2.42 |
| $L = 12, K = 5$ | 0.0084 | 0.013 | 0.020 | 0.059 | 0.68 | 3.53 |
| Exact Pseudo Minv | 0.46 | 0.58 | 1.21 | 2.60 | 11.39 | 39.49 |

are also sampled from $A_{i,j} \sim \mathcal{N}(0, 1/\sqrt{d})$, where $d = \max\{m, \hat{n}\}$. We adopt the SGD to optimize the parameters. Note that our theoretical results in Sec. 4 hold for GD, but they can easily extend to the SGD case. The learning rate is set to 1e-4, and it is reduced by half every 300 iterations. We also fix the batch size as 100, while the number of iteration is set to 1200. *Note that, in the experiments, the adopted exact Pseudo matrix inverse (Minv) is implemented by* ***Pytorch****.*

We adopt the following test loss to measure the performace:

$$\text{Test Loss} := \frac{1}{n} \sum_{i=1}^{n} \left\| \mathbf{A}_i \mathbf{X}_{\{K_{\text{inv}}, i\}} \mathbf{A}_i - \mathbf{A}_i \right\|_F^2,$$

where $\mathbf{X}_{\{K_{\text{inv}}, i\}}$ is the final output of D-Minv or LD-Minv. Here $n = 100$ is the size of test data. With $d = 100$, we first test the influence of $L$ and $K$, which correspond to the order of matrix polynomial and the number of layers in our LD-Minv, respectively. Due to high precision of LD-Minv and D-Minv, for visualization, we take a base-10 logarithm on loss in this experiment.

**Results**. From Fig. 1(a), we can see that the number of layer $K$ plays a more critical role for LD-Minv than the order $L$ of polynomial. When setting $K = 7$, the log of the test error drops from $-1.0$ for $L = 4$ to $-4.6$ for $L = 9$. However, the log of the test error drops from $0.32$ for $K = 2$ to $-4.6$ for $K = 7$, when fixing $L = 9$. Hence, for efficiency, we suggest utilizing large $K$ and small $L$ for LD-Minv.

Compared with D-Minv, according to Fig. 1(b) and Fig. 1(c), LD-Minv can obtain a better test performance even for large $K$ and $L$, although D-Minv converges to Minv extremely fast in this case. Our result demonstrates that learning can help LD-Minv obtain better performance on a problem-dependent data distribution rather than merely fixing the coefficients ahead for time. We notice that LD-Minv and D-Minv obtain a similar precision when $K > 15$, and the reason is that the approximation error for D-Minv is in the order of $1e - 8$ in this case, which lifts very limited space for the improvement by learning.

Fig. 1(b) further verifies the importance of $K$. As shown in our theoretical results (Theorem 1 and Theorem 2); large $K$ indicates smaller generalization bound and higher probability to obtain the linear convergence rate for Algorithm 1.

Another significant advantage of our proposed method is efficiency. D-Minv and LD-Minv can inverse the input matrix with high precision meanwhile with the low computational cost. The benefits of D-Minv and LD-Minv are apparent when the matrix dimension is large. As shown in Table 1, even for large $K$ and $L$, the inference time of LD-Minv is less than a tenth of the time of exact matrix inverse when the size is $1000 \times 1500$. It is well-known that the Minv and matrix multiplication (MatMul) share the same computation complexity theoretically, while their speeds vary a lot numerically. The computation of our method is mostly spent on MatMul, which can be easily accelerated by GPU and is much cheaper than Minv. MatMul and Minv only share the same complexity asymptotically. The constants in the order $\mathcal{O}(\cdot)$ vary significantly, especially when matrix dimension $d \ll \infty$. Moreover, the general MatMul algorithm with a complexity of $\mathcal{O}(d^3)$ isn't optimal, and there exist algorithms that own a complexity of only $\mathcal{O}(d^{2.3})$.

To further investigate the importance of the learning procedure, we conduct the experiments on the matrices with different condition numbers. Given the condition number $\kappa \geq 1$, we generate the data matrix by $\mathbf{A}_i = \mathbf{U}_i \mathbf{S}_i \mathbf{V}_i$, where $\mathbf{U}_i$ and $\mathbf{V}_i$ are randomly sampled from a 100-dimensional

Table 2: $-\log_{10}(\text{Test Loss})$ (the bigger the better) on the test data for LD-Minv and D-Minv with various condition number $\kappa$. D-Minv performs better when $\kappa = 1$. However, LD-Minv shows its strong robustness in the more complex case $\kappa \gg 1$. The network parameters. e.g., $K$ and $L$, mainly determines the final results of LD-Minv rather than the condition number.

| Condition Number $\kappa$ | 1 | 2 | 3 | 4 | 5 | 10 | 50 | 100 | 1000 |
|---|---|---|---|---|---|---|---|---|---|
| D-Minv | 7.64 | 2.83 | 2.49 | 2.36 | 2.30 | 2.27 | 2.26 | 2.26 | 2.25 |
| LD-Minv | 5.95 | 5.01 | 4.19 | 3.62 | 3.49 | 3.30 | 3.21 | 3.22 | 3.21 |

Stiefel manifold. Here $\mathbf{S}_i$ is a diagonal matrix whose diagonal entries are i.i.d. and obey a uniform distribution $\mathcal{U}(1/\kappa, 1)$ with distribution boundaries being $[1/\kappa, 1]$. We let $K = 4, L = 10$ and the batch size be 100. We show the results in Table 2. Not surprisingly, D-Minv suffers from ill-condition problems, e.g., the large condition number deteriorates the performance of D-Minv. However, LD-Minv is not sensitive to the condition number, and it can beat D-Minv easily when the condition number is large. Note that all the results are reported on the unseen data.

### A.1.2 Experimental Results for D-SVD

**Settings.** For D-SVD, we adopt the rectangle matrix to run the experiments. Similarly, our algorithms also work for the square matrix. For the rectangle matrix $\mathbf{A} \in \mathbb{R}^{m\times\hat{n}}$, its elements are also sampled from i.i.d. Gaussian, namely $A_{i,j} \sim \mathcal{N}(0, 1/\sqrt{d})$, where $d = \max\{m, \hat{n}\}$. We let $m = 50$ and $\hat{n} = 100$, and utilize Algorithm 3 to solve our learning problem[5]. Similarly, we also perform the batch training with batch size 30. The learning rate is set to $1e-2$ for D-SVD and $1e-4$ for LD-Minv contained in D-SVD. We do not decay the learning rate during training, and let the number of iterations for D-SVD and LD-Minv be 200 and 100, respectively. For LD-Minv in each layer, we let $L = 4$ and $K = 6$.

Same as Sec. 3, we adopt the following loss to train our D-SVD:

$$\text{Training Loss} := \frac{1}{nK_{\text{svd}}} \sum_{i=1}^{n} \sum_{k=1}^{K_{\text{svd}}} \left( \frac{1}{2} \|\mathbf{\Lambda}_{i,k} - \text{Diag}\left(\mathbf{\Lambda}_{i,k}\right)\|_F^2 - \left\langle \mathbf{M}_i, \mathbf{U}_{i,k}\mathbf{V}_{i,k}^\top \right\rangle \right),$$

where $\mathbf{U}_{i,k}$ and $\mathbf{V}_{i,k}$ are the $k$-th layer's output of D-SVD, and $\mathbf{\Lambda}_{i,k} := \mathbf{U}_{i,k}^\top \mathbf{M}_i \mathbf{V}_{i,k}$. The reason that we introduce the regularization term $\left\|\mathbf{U}_{i,k}^\top \mathbf{M}_i \mathbf{V}_{i,k} - \text{Diag}\left(\mathbf{U}_{i,k}^\top \mathbf{M}_i \mathbf{V}_{i,k}\right)\right\|_F^2$ is the non-uniqueness. In general, any pair $(\mathbf{U}, \mathbf{V}) \in \left\{(\tilde{\mathbf{U}}\mathbf{R}, \tilde{\mathbf{V}}\mathbf{R}) : \mathbf{M} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^\top,\ \mathbf{R}^\top\mathbf{R} = \mathbf{R}\mathbf{R}^\top = \mathbf{I}\right\}$ can maximize the function $\left(\left\langle \mathbf{M}, \mathbf{U}\mathbf{V}^\top \right\rangle\right)$. However, in order to obtain the exact singular vectors of $\mathbf{M}$, the matrix $\left(\mathbf{U}_{i,k}^\top \mathbf{M}_i \mathbf{V}_{i,k}\right)$ should have zero off-diagonal entries. Thus, the regularization term is necessary for the training of D-SVD.

We use the following test loss to report the results:

$$\text{Test Loss} := \frac{1}{n} \sum_{i=1}^{n} \left( \frac{1}{2} \|\mathbf{\Lambda}_{i,K_{\text{svd}}} - \text{Diag}\left(\mathbf{\Lambda}_{i,K_{\text{svd}}}\right)\|_F^2 - \left\langle \mathbf{M}_i, \mathbf{U}_{i,K_{\text{svd}}}\mathbf{V}_{i,K_{\text{svd}}}^\top \right\rangle \right),$$

where $\mathbf{U}_{i,K_{\text{svd}}}$ and $\mathbf{V}_{i,K_{\text{svd}}}$ is the final layer's output of D-SVD, and $\mathbf{\Lambda}_{i,K_{\text{svd}}} := \mathbf{U}_{i,K_{\text{svd}}}^\top \mathbf{M}_i \mathbf{V}_{i,K_{\text{svd}}}$. Note that we also utilize the following quantity to measure the test performance.

$$\text{SVD MSE} := \frac{1}{n} \sum_{i=1}^{n} \left\|\text{Sort}\left(\text{Diag}\left(\mathbf{U}_{i,K_{svd}}^\top \mathbf{M}_i \mathbf{V}_{i,K_{svd}}\right)\right) - \text{SVD}\left(\mathbf{M}_i\right)\right\|_F^2,$$

where $\text{Sort}(\cdot)$ is the operator that sort the input elements in the descending order, $\text{Diag}(\cdot)$ returns the collection of the entries at the diagonal of the input matrix, and $\text{SVD}(\cdot)$ gives out the singular values.

For convenience, we add the mean of the sum of $\text{SVD}(\mathbf{M}_i)$ to the loss in Table 3 to make the loss non-negative. Although increasing $K_{\text{svd}}$ also can improve the performance in the non-learning case,

[5]We replace the GD part by SGD for the generality of the experiment.

Table 3: Training loss and SVD MSE on the test data for D-SVD with various depth $K_{\text{svd}}$. The first two rows are the quantities that measured without training which, to certain extent, are equivalent to the results of the original optimization algorithm (Wen & Yin, 2013; Li et al., 2019). The last two rows are the quantities measured after training on the same test data. The decline of the loss and the MSE is obvious, which demonstrates the matters of learning.

| $K_{\text{svd}}$ | 1 | 5 | 10 | 20 | 30 | 40 | 60 |
|---|---|---|---|---|---|---|---|
| Init Test Loss | 44.9983 | 35.8998 | 26.6607 | 15.6132 | 10.3556 | 7.7057 | 5.0146 |
| Init SVD MSE | 0.8782 | 0.5721 | 0.3373 | 0.1502 | 0.0879 | 0.0568 | 0.0274 |
| Final Test Loss | 23.1838 | 0.9565 | 0.4043 | 0.2694 | 0.2018 | 0.1243 | 0.0305 |
| Final SVD MSE | 0.2528 | 0.0711 | 0.0626 | 0.0167 | 0.0064 | 0.0052 | 0.0029 |

learning can strengthen the role of $K_{\text{svd}}$. Without training, the test loss for $K_{\text{svd}} = 60$ is one eighth of the loss for $K_{\text{svd}} = 1$; in sharp contrast, the reduction can reach one thousandth after training. Due to the introduced regularizer for the training loss, we can obtain a small SVD MSE rather than the small difference between the sum of singular values.

During the training of LD-SVD, we observe that our results are robust to the setting of $K$ and $L$. In general, the forward process can converge even with very inexact matrix inverse (Li et al., 2019). Moreover, our introduced learnable step size $\{\tilde{t}_{U_k}, \tilde{t}_{V_k}\}$ further provides the freedom to adjust the update direction for $\mathbf{U}_k$ and $\mathbf{V}_k$. Thus, for efficiency, a relative small $K$ and $L$ is enough for each layer's LD-Minv.

### A.1.3 Plug-and-Play: Differentiable Non-blind Deconvolution

Non-blind deconvolution (NbD) aims to restore the latent image $\mathbf{z}$ from corrupted observation $\mathbf{y}$ with known blur kernel $\mathbf{b}$. In this experiment, we consider a well-known sparse coding formulation:

$$\mathbf{y} = \mathbf{b} \otimes \mathbf{z} + \mathbf{n} = \mathbf{B}\mathbf{W}^\top \mathbf{x} + \mathbf{n},$$

where $\otimes$ is the convolution operator, $\mathbf{x}$ and $\mathbf{n}$ are the sparse code and unknown noises, respectively. $\mathbf{B}$ is the matrix form of kernel $\mathbf{b}$, and $\mathbf{W}^\top$ is the inverse of the wavelet transform $\mathbf{W}$ (i.e., $\mathbf{x} = \mathbf{W}\mathbf{z}$ and $\mathbf{z} = \mathbf{W}^\top\mathbf{x}$). We utilize the following optimization to perform the non-blind deconvolution:

$$\min_{\mathbf{x},\mathbf{z}} g(\mathbf{x},\mathbf{z}) := \frac{1}{2}\|\mathbf{y} - \mathbf{B}\mathbf{z}\|_2^2 + \lambda\|\mathbf{x}\|_1, \quad \text{s.t.} \quad \mathbf{z} = \mathbf{W}^\top\mathbf{x},$$

where $\lambda > 0$ is the balance parameter. A usual way to solve this problem is linearized ADMM (L-ADMM) (Lin et al., 2011), read as:

$$\begin{cases} \mathbf{a}_k = \mathbf{z}_k - \alpha_k\big(\boldsymbol{\lambda}_k + \beta(\mathbf{z}_k - \mathbf{W}^\top\mathbf{x}_k)\big), & \mathbf{z}_{k+1} = \big(\alpha_k\mathbf{B}\mathbf{B}^\top + \mathbf{I}\big)^{-1}\big(\mathbf{a}_k + \alpha_k\mathbf{B}^\top\mathbf{y}\big), \\ \mathbf{b}_k = \mathbf{x}_k - \gamma_k\mathbf{W}^\top\big(\boldsymbol{\lambda}_k + \beta(\mathbf{z}_{k+1} - \mathbf{W}^\top\mathbf{x}_k)\big), & \mathbf{x}_{k+1} = \operatorname{sgn}(\mathbf{b}_k) \odot \max\Big(\mathbf{b}_k - \dfrac{\gamma_k}{\lambda}, 0\Big), \\ \boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \beta\big(\mathbf{z}_{k+1} - \mathbf{W}^\top\mathbf{x}_{k+1}\big), \end{cases} \tag{13}$$

where $\alpha_k > 0, \gamma_k > 0$ and $\beta > 0$ are penalty parameters, $\odot$ is the Hadamard product, and $\boldsymbol{\lambda}$ is the Lagrange multiplier.

**Differentiable NbD.** Inspired by the Eq. (13), we can easily obtain the learning-based non-blind deconvolution method:

$$\begin{cases} \mathbf{a}_k = \mathbf{z}_k - \alpha_k\widetilde{\mathbf{I}}_k\big(\boldsymbol{\lambda}_k + \beta(\mathbf{z}_k - \mathbf{W}^\top\mathbf{x}_k)\big), & \mathbf{T}_k = \text{LD-Minv}\left(\alpha_k\mathbf{B}\mathbf{B}^\top + \mathbf{I}, \mathbf{C}_k\right) \\ \mathbf{z}_{k+1} = \mathbf{T}_k\big(\mathbf{a}_k + \alpha_k\mathbf{B}^\top\mathbf{y}\big), & \\ \mathbf{b}_k = \mathbf{x}_k - \gamma_k\widetilde{\mathbf{W}}_k\big(\boldsymbol{\lambda}_k + \beta(\mathbf{z}_{k+1} - \mathbf{W}^\top\mathbf{x}_k)\big), & \mathbf{x}_{k+1} = \text{ReLU}\Big(\mathbf{b}_k - \dfrac{\gamma_k}{\lambda}\Big) - \text{ReLU}\Big(-\mathbf{b}_k - \dfrac{\gamma_k}{\lambda}\Big), \\ \boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \beta\big(\mathbf{z}_{k+1} - \mathbf{W}^\top\mathbf{x}_{k+1}\big), \end{cases} \tag{14}$$

where $\mathcal{S}_{\text{NbD}} := \{\widetilde{\mathbf{I}}_k, \widetilde{\mathbf{W}}_k, \alpha_k, \gamma_k, \beta\}_{k=1}^{K_{\text{NbD}}}$ is the collection of all learnable parameters, $\widetilde{\mathbf{I}}_k$ and $\widetilde{\mathbf{W}}_k$ are introduced learnable matrices (in the implementation, we choose them as the convolution layer

**Algorithm 4** Joint Training for D-NbD and LD-Minv

```
Input: Training data $\{\mathbf{y}_i\}_{i=1}^n$, number of iterations $T_{\text{NbD}}$ and $T_{\text{inv}}$, step sizes $\eta_{\text{NbD}}$ and $\eta_{\text{inv}}$.
1: Set $\mathbf{x}_{i,0} = \mathbf{y}_i, \mathbf{z}_{i,0} = \mathbf{W}^\top \mathbf{x}_{i,0}$ and $\boldsymbol{\lambda}_0 = \mathbf{0}$.
2: for $t = 1, \cdots, T_{\text{svd}}$ do
3:     Fix the learnable parameters $\mathcal{S}_{\text{NbD}}^{(t)}$ and train each layer's LD-Minvs by Algorithm 1.
4:     Update $\mathcal{S}_{\text{NbD}}^{(t+1)} = \mathcal{S}_{\text{NbD}}^{(t)} - \eta_{\text{NbD}} \cdot \mathcal{D}_{[\mathcal{S}_{\text{NbD}}^{(t)} = \mathcal{S}_{\text{NbD}}]} \mathcal{Q}_n$, where $\mathcal{Q}_n$ is presented in Eq.(15).
5: end for
Output: $\{\mathbf{z}_{i,k}\}_{i=1,k=1}^{n,K_{\text{NbD}}}$.
```

Table 4: Averaged PSNR and Inference Time(s) on the test images randomly selected from Schmidt & Roth (2014). Here fL-ADMM denote the L-ADMM given in Eq. (13) with fixed $\alpha_k, \gamma_k$ for all $k$, aL-ADMM represents the L-ADMM with adaptive $\alpha_k = \max\{0.98^k \alpha_0, \epsilon\}, \gamma_k = \max\{0.98^k \gamma_0, \epsilon\}$ with $\epsilon$ =1e-4. When D-NbD in Eq. (14) utilizes D-Minv and LD-Minv to perform Minv during the forward propagation, we denote them by D-NbD(D-Minv) and D-NbD(LD-Minv), respectively.

| Noise level | Metric | fL-ADMM | aL-ADMM | D-NbD(D-Minv) | D-NbD(LD-Minv) |
|---|---|---|---|---|---|
| 1% | PSNR | 26.13 | 27.50 | 28.97 | **30.00** |
| | Time(s) | 7.83 | 21.48 | 1.27 | **1.20** |
| 2% | PSNR | 24.06 | 25.33 | 27.82 | **28.11** |
| | Time(s) | 5.26 | 14.35 | 1.28 | **1.19** |
| 3% | PSNR | 23.38 | 24.67 | 26.94 | **27.13** |
| | Time(s) | 5.72 | 13.90 | 1.28 | **1.20** |

with compatible size), $\mathbf{C}_k$ is the learnable parameters for each layer's LD-Minv, and $\text{ReLU}(a) = \max\{a, 0\}$ is the Rectified Linear Unit (ReLU) function. Note that

$$\text{sgn}(\mathbf{b}) \odot \max(\mathbf{b} - \gamma, 0) = \text{ReLU}(\mathbf{b} - \gamma) - \text{ReLU}(-\mathbf{b} - \gamma).$$

Eq. (14) can be implemented by a DNN. Given the training images $\{\mathbf{y}_i\}_{i=1}^n$, we adopt the following loss to train our D-NbD:

$$\mathcal{Q}_n(\mathcal{S}_{\text{NbD}}) := \frac{1}{nK_{\text{NbD}}} \sum_{i=1}^{n} \sum_{k=1}^{K_{\text{NbD}}} \left( g(\mathbf{x}_{i,k}, \mathbf{W}^\top \mathbf{x}_{i,k}) \right), \tag{15}$$

where $\mathbf{x}_{i,k}$ is the $k$-th layer's output of D-NbD. The training process for D-NbD is given in Algorithm 4.

**Settings.** The training and test images come from Schmidt & Roth (2014), in which 300 (random selected) have been used for training and the others are used for test. We add different levels of the Gaussian noise to generate our corrupted observations $\{\mathbf{y}_i\}_{i=1}^n$. We set the patch size as $160 \times 160$, and let the batch size be 15. Finally, Adam is adopted and executed 1000 epochs for learning rate ranging from 1e-4 to 1e-6 (it is reduced by half every 140 iterations). The observations $\{\mathbf{y}_i\}$ are corrupted by blur kernel with size ranging from $17 \times 17$ to $57 \times 57$ and the levels of Gaussian noise varying from $1\%$ to $3\%$.

**Results.** From Table 4, we find that fL-ADMM is much more efficient than aL-ADMM. fL-ADMM does not need to perform the inverse balbala since $\alpha_k, \gamma_k$ are fixed, which however is also why fL-ADMM performs poorly. It can not find the universal $\alpha_k, \gamma_k$ for all test images. aL-ADMM can perform better with adaptive step size, but it needs exact Minv in each iteration. Compared to them, D-NbD outperforms L-ADMM by a large margin in terms of speed and performance. Note that, for one iteration, the computation complexity of fL-ADMM is much lower than the other three since it only involves the MatMul. However, it consumes more than 600 iterations to obtain a relatively good solution. In fact, just comparing the Minv time in one iteration for aL-ADMM and D-NbD, our methods' time is only almost half of that of exact Minv in PyTorch. Fortunately, the learnability of D-NbD helps us obtain a better or equally good solution using one order-of-magnitude fewer iterations. Hence, the total time can only be one-tenth of that of aL-ADMM. Moreover, the learnability of LD-Minv further introduces more flexibility, which allows D-NbD(LD-Minv) to achieve better results within fewer layers. Thus, D-NbD(LD-Minv) is more efficient due to less calculation.

### A.2 Discussion

Not only D-NbD, based on the LD-Minv and D-SVD, but there are also rich potential applications, including image recovery or denoising, numerical algorithm acceleration, blind deconvolution, sparse coding, subspace clustering, etc. All these compelling applications may be beyond this paper's scope, and we leave them as future work. Moreover, due to the differentiability, D-SVD can help us design more abundant singular-value-related training loss for DNN, which is impossible in the previous non-differentiable situation.

## B Notation

Positive definite matrix $\mathbf{A}$ is denoted by $\mathbf{A} \succ 0$. Transpose of the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is $\mathbf{A}^\top$. We denote by $\mathbf{A}^{-1}$ and $\mathbf{A}^\dagger$ the matrix inverse and Moore–Penrose inverse, respectively. The Euclidean inner product between two matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times n}$ is defined as $\langle \mathbf{A}, \mathbf{B}\rangle := \mathrm{Tr}\left(\mathbf{A}^\top \mathbf{B}\right)$, where $\mathrm{Tr}(\cdot)$ is the trace of a matrix. $\kappa(\mathbf{A}) := \|\mathbf{A}^\dagger\|\|\mathbf{A}\|$ is the generalized condition number of a matrix, where $\|\cdot\|$ is the spectral norm. We denote by $\sigma(\mathbf{A})$ the singular values vectors (no need to be ordered) of the matrix $\mathbf{A}$. Similarly, we denote by $\sigma_+(\mathbf{A})$ the positive singular values vectors.

Let $\|\cdot\|_F$ be Frobenius norm. Given a differentiable function the first and second derivative of $\mathcal{F}$ at $\mathbf{Y}$ are denoted by: linear operator $\mathcal{D}_{[\mathbf{X}=\mathbf{Y}]}\mathcal{F}(\cdot) : \mathbb{R}^m \to \mathbb{R}^n := \left(\frac{\partial \mathcal{F}_i(\mathbf{Y})}{\partial X_j}\right)(\cdot)$ and quadratic operator $\mathcal{D}^2_{[\mathbf{X}=\mathbf{Y}]}\mathcal{F}(\cdot,\cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^n := \left(\frac{\partial^2 \mathcal{F}_i(\mathbf{Y})}{\partial X_j \partial X_k}\right)(\cdot,\cdot)$, respectively, where $X_i$ is the $i$-th entry of $\mathbf{X}$. We also denote by $\mathcal{F}^{-1}$ the inverse of the function $\mathcal{F}$ and let the composition map $f \circ g(x) := f(g(x))$. $\odot$ is the Hadamard product and $[K] = \{1, 2, \cdots, K\}$.

## C Additional Theoretical Results

In this section, we provide the additional theoretical results of D-Minv and D-SVD. We start by a warm-up case for D-Minv.

### C.1 Results for LD-Minv

#### C.1.1 Warm-up: One Parameter for One Layer

As a warm-up case, we consider only to learn one coefficient for each layer, as a special case of learnable D-Minv:

$$\mathbf{X}_{k+1} = \mathbf{X}_k \left(\sum_{l=0}^{L-1} \mathbf{E}_k^l + C_k \mathbf{E}_k^L\right), \quad \mathbf{E}_k = \mathbf{I} - \mathbf{A}\mathbf{X}_k, \tag{16}$$

where $C_k \in \mathbb{R}$ are the learnable coefficients. By setting $\mathbf{X}_0 = \frac{1}{c_0}\mathbf{A}^\top$, where $c_0 > \|\mathbf{A}\|^2$, and let $\mathbf{C}_k = 0$ for all $k$, by Lemma 1, we know that $\left\|\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{X}_k\right\| = e_0^{L^k}$. One important question is if we let the coefficient of $\mathbf{E}_k^L$ to be learnable, by training, will D-Minv obtain a faster convergence rate?

In general, previous works show that learning-based methods share the same convergence order with their fixed version but with a better statistical constant. Interestingly, beyond the constant, we found that D-Minv can improve the order of convergence by learning.

**Lemma 2** (Learnability Matters). *Assume these is only one training data $\mathbf{A}$. Provided that $\mathbf{X}_0 = \frac{1}{c_0}\mathbf{A}^\top$, where $c_0 > \|\mathbf{A}\|^2$, then we have:*

$$e_{k+1} = (1 - C_k)\cdot e_k^L + C_k \cdot e_k^{L+1}, \quad \textit{where} \quad e_k = \left\|\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{X}_k\right\|, \quad \textit{and} \quad k \in [K], L \geq 4.$$

*Moreover, the gradient of $\mathcal{L}$ w.r.t. $C_k$ is negative, i.e., $\mathcal{D}_{[C_k]}\mathcal{L} < 0$ for all $C_k \in [0, 1+\epsilon]$, where $\epsilon > 0$ depends on $e_{k-1}$. Additionally, $\mathcal{D}_{[C_k]}\mathcal{L} \approx 0$ for $k \in [K-1]$ when $C_k \in [1-\epsilon, 1+\epsilon]$.*

Starting from 0, according to the gradient's negativeness, the coefficients $C_k$ for $k \in [K-1]$ will converge to the range $[1-\epsilon, 1+\epsilon]$ when adopting a proper step size for GD. In this case, i.e., $C_k$

are around 1 for all $k$, the sequence $\{\mathbf{X}_k\}$ converges to the Moore–Penrose inverse in $\tilde{L}$-th order, where $L < \tilde{L} \leq L+1$. In other words, by learning from one data, D-Minv finds a way to improve the order of convergence for any input. Remarkably, $C_k = 1$ for all $k$ is not a global or even local minimum for the training loss $\mathcal{L}$. For the learnable coefficient $C_K$ at the last layer, satisfying its first order condition usually implies obtaining the exact solution, i.e., $e_{K+1} = 0$. In summary, not only improve the convergence speed, learnable D-Minv makes the exact fitting come true.

### C.1.2 Discussion for Assumption 1

In general, the bounded assumption for the singular values the is weak and easy to satisfy. For example, singular values of the matrices with i.i.d. zero mean and unit variance entries asymptotically obey the Quadrant Law (Dozier & Silverstein, 2007; Shen, 2001), i.e., $\sigma \sim \frac{1}{\pi}\sqrt{4-\sigma^2}$, $\sigma \in [0,2]$, which implies the singular values have an upper bound. On the other hand, a well-known fact from the random matrix theory is a zero mean and unit variance random matrix is non-singular with high probability. Even for the square matrix $\mathbf{A}$, with probability at least $1-\delta$, we still we have $\sigma_{\min}(\mathbf{A}) \geq \delta d^{-1/2}$ (Rudelson & Vershynin, 2008), where $\sigma_{\min}(\mathbf{A})$ is the smallest singular value of the matrix $\mathbf{A}$.

### C.1.3 Lipschitz Smoothness before Generalization

We would also like to remark that our generalization result can easily be extended to the stochastic case. The proof is the same as the Lemma 4.3 in Ji & Telgarsky (2019) or Theorem 3.3 in Cao & Gu (2019).

In general, smoothness plays an important role in the generalization analysis. The covering number of a smooth function usually is small. We start by showing the local smoothness of learnable D-Minv.

**Proposition 2** (Local Lipschitz smoothness)**.** *Define a set of coefficients collection:*

$$\mathcal{C}^* := \left\{\mathbf{C} \mid \forall\, C_{\{k,l\}} \in \mathbf{C},\ \left|\mathbf{C}_{\{k,l\}} - 1\right| \leq \frac{1}{8}\right\},\quad (k,l) \in [K]\times[L],\quad L \geq 4.$$

*Let $\mathbf{P_A} = \mathbf{A}\mathbf{A}^\dagger$. Given any coefficients collection $\mathbf{C} \in \mathcal{C}$, we denote by $\mathcal{G}_k(\cdot)$ the map from $\mathbf{P_A}\mathbf{E}_k$ to $\mathbf{P_A}\mathbf{E}_{k+1}$ in (3), i.e., $\mathcal{G}_k(\mathbf{E}) := \mathbf{P_A} - \mathbf{P_A}(\mathbf{I}-\mathbf{E})\left(\sum_{l=0}^{L-1} C_{\{k,l\}}\mathbf{E}^l\right)$, and let $\mathcal{H}_k(\cdot) := \mathcal{R}\circ\mathcal{G}_{K-1}\circ\cdots\circ\mathcal{G}_k(\cdot)$, where $\mathcal{R}(\mathbf{E}) := \sum_{\hat{k}=k}^{K}\left\|\mathbf{W}_{\hat{k}}\mathbf{E}_{\hat{k}}\mathbf{A}\right\|_F^2$. Suppose $\|\mathbf{A}\|_2 \leq 1$ and $\forall k\ \|\mathbf{W}_k\| = \mathcal{O}(1)$, then we have:*

$$\left|\mathcal{H}_k\left(\widehat{\mathbf{E}}\right) - \mathcal{H}_k\left(\widetilde{\mathbf{E}}\right)\right| \leq 2\left\|\mathbf{P_A}\widehat{\mathbf{E}} - \mathbf{P_A}\widetilde{\mathbf{E}}\right\|_*,\quad \forall k \in [K],$$

*where $\widehat{\mathbf{E}}$ and $\widetilde{\mathbf{E}}$ are with the same size and same norm upper bound as $\mathbf{E}_k$, $d$ is the dimension of positive singular value vector $\sigma_+(\mathbf{A})$, and $\|\cdot\|_*$ is the matrix nuclear norm.*

In general, the composition of two Lipschitz continuous functions leads to a worse Lipschitz constant. However, our LD-Minv shares a consistent constant for all layers. This is important since the layer-wisely local smoothness usually implies a small covering number of the whole D-Minv (Bartlett et al., 2017; Wei & Ma, 2019), which results in a tight generalization bound.

## C.2 Results for D-SVD

We characterize the generalization performance of D-SVD trained by GD in this theorem. First of all, we present several definitions. Let

$$\mathcal{M}_n(\tilde{\mathbf{t}}, \mathbf{C}) := \frac{1}{nK_{\text{svd}}}\sum_{i=1}^{n}\sum_{k=1}^{K_{\text{svd}}}\left\langle \mathbf{M}_i, \mathbf{U}_{i,k}\mathbf{V}_{i,k}^\top\right\rangle,$$

where $(\tilde{\mathbf{t}}, \mathbf{C})$ are the collection of the parameters of D-SVD and the LD-Minvs in each layer. Denote by $f(\mathbf{U} \mid \mathbf{V}, \mathbf{M}, \tilde{t}_k, \mathbf{C}_k)$ the operations in Eq. (12) that map $\mathbf{U}_k$ to $\mathbf{U}_{k+1}$, i.e.,

$$\mathbf{U}_{k+1} := f(\mathbf{U}_k \mid \mathbf{V}_k, \mathbf{M},\ \tilde{t}_k, \mathbf{C}_k), \tag{17}$$

where $(\tilde{t}_k, \mathbf{C}_k) \in \mathbb{R} \times \mathbb{R}^{LK_{\text{inv}}}$ are the learnable parameters of D-SVD and LD-Minv at the $k$-th layer of D-SVD. We define the coefficients set:

$$\mathcal{C}_k := \Big[-\frac{\varepsilon_u}{2}, +\frac{\varepsilon_u}{2}\Big] \times \Big\{\mathbf{C} : \Big\|\mathbf{C} - \mathbf{C}^{(0)}\Big\|_F = \widetilde{\mathcal{O}}\Big(\frac{1}{\sqrt{K_{\text{inv}}}}\Big)\Big\},$$

where

$$\varepsilon_u := \min_i \frac{1}{\|\mathbf{P}_i\|}, \quad \text{in which} \quad \mathbf{P}_i = \mathbf{M}_i \mathbf{V}_k \mathbf{U}_k^\top - \mathbf{U}_k \mathbf{V}^\top \mathbf{M}_i^\top, \quad \forall i \in [n],$$

**Proposition 3** (Covering Number of Single Layer). *Denote $\mathcal{F}$ the class of function in Eq. (17), i.e.,*

$$\mathcal{F}_k := \big\{f(\mathbf{U}_k \mid \mathbf{V}_k, \mathbf{M}, \tilde{t}_k, \mathbf{C}_k) : (\tilde{t}_k, \mathbf{C}_k) \in \mathcal{C}_k\big\}.$$

*Then we have:*

$$\ln \mathcal{N}(\mathcal{F}_k, \epsilon, \|\cdot\|)) = \mathcal{O}\Big(LK_{\text{inv}} \ln\Big(\frac{1}{\epsilon\sqrt{K_{\text{inv}}}}\Big) + \ln\Big(\frac{\|\mathbf{M}\|}{\epsilon}\Big)\Big).$$

**Theorem 3** (Generalization Bound for D-SVD). *Denote by $\mathcal{P}_{\mathbf{M}}$ the distribution of $\mathbf{M}_i$ and let $d := \max\{m, \hat{n}\}$. Suppose that $\|\mathbf{M}_i\| \leq 1$, $\forall i \in [n]$ and $K_{\text{inv}} = \Omega\big(\max\{nd, K_{\text{svd}}^2\}\big)$, then with probability at least $1 - \delta$, the iterate $\tilde{\mathbf{t}}^{(t)}$ of Algorithm 3 holds that:*

$$\mathop{\mathbb{E}}_{\mathbf{M} \sim \mathcal{P}_{\mathbf{M}}} \Big[\mathcal{M}_n\Big(\tilde{\mathbf{t}}^{(t)}\Big)\Big] \leq \mathcal{M}_n\Big(\tilde{\mathbf{t}}^{(t)}\Big) + \widetilde{\mathcal{O}}\Bigg(\sqrt{\frac{K_{\text{svd}}^3 L}{n}}\Bigg) + \mathcal{O}\Bigg(\sqrt{\frac{\ln(1/\delta)}{n}}\Bigg),$$

*for $t = 0, 1, \cdots, T_{\text{svd}}$, where $\mathbb{E}_{\mathbf{M} \sim \mathcal{P}_{\mathbf{M}}}[\mathcal{M}_n]$ is the expected value of $\mathcal{M}_n$ under the probability measure $\mathcal{P}_{\mathbf{M}}$ and $\widetilde{\mathcal{O}}(\cdot)$ hides log factors.*

## D Omitted Proof for Fixed D-Minv in Section 2

### D.1 Proof of Lemma 1

We first verify that $e_0 < 1$, suppose $\operatorname{rank}(\mathbf{A}) = r$.

$$e_0 = \big\|\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{X}_0\big\| = \Big\|\mathbf{U}\mathbf{U}^\top - \frac{1}{c}\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top\Big\| = \Big\|\mathbf{I}_r - \frac{1}{c}\mathbf{\Sigma}^2\Big\| < 1,$$

where $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, $\mathbf{I}_r \in \mathbb{R}^{r \times r}$ is the identity matrix, and the last inequality comes from the fact $c > \frac{1}{2}\|\mathbf{A}\|^2$. Form Eq. (2), we have

$$\begin{aligned}
\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{X}_{k+1} &= \mathbf{A}\mathbf{A}^\dagger(\mathbf{I} - \mathbf{A}\mathbf{X}_{k+1}) = \mathbf{A}\mathbf{A}^\dagger\left(\mathbf{I} - \mathbf{A}\mathbf{X}_k\left(\sum_{l=0}^{L-1}\mathbf{E}_k^l\right)\right) \\
=& \mathbf{A}\mathbf{A}^\dagger\left(\mathbf{I} - (\mathbf{I} - \mathbf{E}_k)\left(\sum_{l=0}^{L-1}\mathbf{E}_k^l\right)\right) = \mathbf{A}\mathbf{A}^\dagger\left(\mathbf{I} - \sum_{l=0}^{L-1}\mathbf{E}_k^l + \sum_{l=0}^{L-1}\mathbf{E}_k^{l+1}\right) \\
=& \mathbf{A}\mathbf{A}^\dagger\big(\mathbf{E}_k^L\big) = \mathbf{A}\mathbf{A}^\dagger(\mathbf{I} - \mathbf{A}\mathbf{X}_k)^L = \big(\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{X}_k\big)^L,
\end{aligned}$$

i.e.,

$$\big\|\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{X}_k\big\| = \big\|\mathbf{A}\mathbf{A}^\dagger - \mathbf{A}\mathbf{X}_{k-1}\big\|^L = e_0^{L^k}.$$

We finish the proof.

### D.2 Proof of Proposition 1

Note that for any matrix $\mathbf{X}$, the matrix polynomial of $\mathbf{X}$ share the same column and row space with $\mathbf{X}$. In general, the iterative equation (2) does not change the column and row space from the begining. We finish the proof by induction. The conclusion is obvious for $\mathbf{X}_0$ due to $\mathbf{X}_0 = \frac{1}{c_0}\mathbf{A}^\top$. We assume the proposition holds for $\mathbf{X}_k$, then:

$$\mathbf{A}^\dagger\mathbf{A}\mathbf{X}_{k+1} = \mathbf{A}^\dagger\mathbf{A}\mathbf{X}_k\left(\sum_{l=0}^{L-1}\mathbf{E}_k^l\right) = \mathbf{X}_k\left(\sum_{l=0}^{L-1}\mathbf{E}_k^l\right) = \mathbf{X}_{k+1},$$

and

$$\mathbf{X}_{k+1}\mathbf{A}\mathbf{A}^{\dagger} = \mathbf{X}_k\left(\sum_{l=0}^{L-1}\mathbf{E}_k^l\right)\mathbf{A}\mathbf{A}^{\dagger} = \mathbf{X}_k\left(\sum_{l=0}^{L-1}(\mathbf{I}-\mathbf{A}\mathbf{X}_k)^l\right)\mathbf{A}\mathbf{A}^{\dagger}$$
$$=\mathbf{X}_k\mathbf{A}\mathbf{A}^{\dagger}\left(\sum_{l=0}^{L-1}(\mathbf{I}-\mathbf{A}\mathbf{X}_k)^l\right) = \mathbf{X}_{k+1},$$

Hence, we can conclude that:

$$\mathbf{A}^{\dagger}\mathbf{A}\mathbf{X}_k = \mathbf{X}_k, \quad \mathbf{X}_k\mathbf{A}\mathbf{A}^{\dagger}, \quad \forall k.$$

We finish the proof.

## E Omitted Proof for Learnable D-Minv in Sections 4 and C

In the theoretical part, we consider a more general training regression loss:

$$\mathcal{L}_n(\mathbf{C}) := \frac{1}{2nK}\sum_{i=1}^{n}\sum_{k=1}^{K}\left\|\mathbf{V}_{\{k,i\}}\left(\mathbf{A}_i\mathbf{X}_{\{k,i\}}\mathbf{A}_i - \mathbf{A}_i\right)\right\|_F^2, \tag{18}$$

where $\mathbf{C} = \{C_{\{k,l\}},\ (k, l+1) \in [K]\times[L]\}$ is the collection of all learnable parameters, $\mathbf{V}_{\{k,i\}}$ is a diagonal matrix whose diagonal entries are i.i.d. and obey a zero mean bounded distribution (w.l.o.g. we let the bound be $\pm\tilde{b}_v$).

Note that $\mathbf{V}_{\{k,i\}}$'s are fixed during training and testing. Introducing $\mathbf{V}_{\{k,i\}}$ has two advantages. One is re-weighting the error term. As we can see from the fixed D-Minv, the error term decays exponentially. In that way, the smallest positive singular value dominate the gradient flow. However, for LD-Minv, we hope that the gradient flow comes from all positive singular values' loss during training. The random re-weighting strategy is a good choice for numerical calculation when the order of singular values is unknown. Another advantage is that the introduced $\mathbf{V}_{\{k,i\}}$s, with very high probability (see Claim 1 in Sec. E.2), make the landscape of the training loss locally strongly convex, which is an excellent benefit from the optimization perspective.

Notably, by setting the diagonal entries of $\mathbf{V}_{\{k,i\}}$ as the independent symmetric Bernoulli random variables, the training loss here in Eq. (18) will degenerate into the loss given in Eq. (4). Hence, all the theoretical results in this section proven for Eq. (18) also hold for Eq. (4).

It is worth mentioning that the weight matrices $\{\mathbf{V}_{\{k,i\}}\}$ here has nothing to do with the learned singular vectors $\{\mathbf{V}_{i,k}\}$ for D-SVD. **Please distinguish them**.

### E.1 General Analysis

Before start the proof, we first make some general analysis. It is easy to see that Proposition 1 also holds for learnable D-Minv. Thus, we can observe that:

$$\mathbf{X}_k\mathbf{E}_k^l = \mathbf{X}_k\mathbf{A}\mathbf{A}^{\dagger}\mathbf{E}_k^l, \quad \forall (k,l).$$

Hence, we obtain an equivalent form of Eq. (3):

$$\mathbf{X}_{k+1} = \mathbf{X}_k\left(\sum_{l=0}^{L-1}C_{\{k,l\}}\widetilde{\mathbf{E}}_k^l\right), \quad \widetilde{\mathbf{E}}_k = \mathbf{A}\mathbf{A}^{\dagger} - \mathbf{A}\mathbf{X}_k, \quad L \geq 4, \tag{19}$$

Due to the invariance of the row and column space, we can only concern on the singular values vectors and rewrite Eq. (19) as :

$$\mathbf{x}_{k+1} = \mathbf{x}_k \odot \left(\sum_{l=0}^{L-1}C_{\{k,l\}}\mathbf{e}_k^l\right), \quad \mathbf{e}_k = \mathbf{1} - \mathbf{a}\odot\mathbf{x}_k, \quad L \geq 4, \tag{20}$$

where $\mathbf{1} \in \mathbb{R}^d$ is the all-one vector ($d$ is the dimension of the positive singular vector $\sigma_+(\mathbf{A})$), and:

$$\mathbf{a} = \sigma_+(\mathbf{A}), \quad \mathbf{x}_k = \sigma_+(\mathbf{X}_k), \quad \mathbf{e}_k = \sigma_+\left(\widetilde{\mathbf{E}}_k\right), \quad \mathbf{e}_k^l = \underbrace{\mathbf{e}_k \odot \cdots \odot \mathbf{e}_k}_{l}\ ; \tag{21}$$

recall that $\sigma(\mathbf{A})$ the singular values vectors (no need to

**Remark 4.** *We only utilize Eq. (19), the equivalent form of Eq. (3), in this section for the convenience of theoretical analysis. For implementation, we still use Eq. (3) and do **NOT** calculate the pseudo inverse $\mathbf{A}^\dagger$ throughout learning.*

Now we continue the derivation:

$$\begin{aligned}\mathbf{e}_{k+1} = \mathbf{1}-\mathbf{a}\odot\mathbf{x}_{k+1} &= \mathbf{1}-\mathbf{a}\odot\mathbf{x}_k\odot\left(\sum_{l=0}^{L-1}C_{\{k,l\}}\mathbf{e}_k^l\right)\\ &= \mathbf{1}-(\mathbf{1}-\mathbf{e}_k)\odot\left(\sum_{l=0}^{L-1}C_{\{k,l\}}\mathbf{e}_k^l\right) = \mathbf{1}-\left(\sum_{l=0}^{L-1}C_{\{k,l\}}\mathbf{e}_k^l-\sum_{l=0}^{L-1}C_{\{k,l\}}\mathbf{e}_k^{l+1}\right)\\ &= \mathbf{e}_k^L+(1-C_{k,0})\mathbf{1}+\left(\sum_{l=1}^{L-1}\left(C_{\{k,l-1\}}-C_{\{k,l\}}\right)\mathbf{e}_k^l\right)+\left(C_{\{k,L-1\}}-1\right)\mathbf{e}_k^L\\ &= \mathbf{e}_k^L+\widehat{\mathbf{E}}_k\mathbf{c}_k,\end{aligned} \tag{22}$$

where $\mathbf{e}_k^0=\mathbf{1}$,

$$\widehat{\mathbf{E}}_k\in\mathbb{R}^{d\times(L+1)} := \left[\mathbf{e}_k^0;\mathbf{e}_k^1;\cdots;\mathbf{e}_k^L\right], \tag{23}$$

and

$$\mathbf{c}_k\in\mathbb{R}^{L+1} := \left[1-C_{k,0},C_{\{k,0\}}-C_{\{k,1\}},\cdots,C_{\{k,L-2\}}-C_{\{k,L-1\}},C_{\{k,L-1\}}-1\right]^\top. \tag{24}$$

We define:

$$\ell(\mathbf{C},\mathbf{A}) := \frac{1}{2}\sum_{k=1}^{K}\|\mathbf{V}_k(\mathbf{A}\mathbf{X}_k\mathbf{A}-\mathbf{A})\|_F^2 = \frac{1}{2}\left\|\sum_{k=1}^{K}\mathbf{v}_j\odot\left(\mathbf{a}^2\odot\mathbf{x}_k-\mathbf{a}\right)\right\|_2^2 = \frac{1}{2}\sum_{j=1}^{K}\|\mathbf{v}_k\odot\mathbf{a}\odot\mathbf{e}_k\|_2^2, \tag{25}$$

where $\mathbf{C}=\{C_{\{k,l\}},\ (k,l+1)\in[K]\times[L]\}$ is the collection of all learnable parameters for D-Minv, $\mathbf{V}_k$ is a diagonal matrix whose diagonal entries $\mathbf{v}_k=\operatorname{diag}(\mathbf{V}_k)$ are i.i.d. and obey a zero mean bounded distribution (w.l.o.g. we let the bound be $\pm\bar{b}_v$), and $\|\cdot\|_2$ is the vector $\ell_2$ norm; recall the definitions of $\mathbf{a}$, $\mathbf{e}_k$ and $\mathbf{x}_k$ from Eq. (21). Note that $\mathbf{v}_k,\ \forall k$ is fixed during training.

From Lemma 1, we know that $\left\|\mathbf{A}\mathbf{A}^\dagger-\mathbf{A}\mathbf{X}_k\right\|$ decays extremely fast. We can run fixed D-Minv for several times before training learnable D-Minv. In this case, $\mathbf{X}_0$ is the output of the $\widetilde{K}$-layer fixed D-Minv. Hence, without loss of generality (w.l.o.g.), we suppose the following assumption holds in this section.

**Assumption 2** (Well-Bounded $\mathbf{E}_0$). *Assume* $\left\|\widetilde{\mathbf{E}}_0\right\|=\|\mathbf{e}_0\|_\infty\le\frac{1}{2}$.

We first provide several propositions of learnable D-Minv.

**Proposition 4** (Upper Bound for Perturbation). *Suppose that* $\left|C_{\{k,l\}}-1\right|=\delta<\frac{1}{8},\ \forall\,(k,l+1)\in[K]\times[L]$*, then we have:*

$$\left\|\widetilde{\mathbf{E}}_k\right\|=\|\mathbf{e}_k\|_\infty\le e_0^{L^k}+\frac{17}{8}\delta,\quad\forall k\in[K].$$

*Proof.* We first show that $\|\mathbf{e}_k\|_\infty<\frac{1}{2}$ for all $k$ by induction. By Eq. (22), for $k=0$, we know that:

$$\|\mathbf{e}_1\|_\infty\le\|\mathbf{e}_0\|_\infty^L+\left\|\widehat{\mathbf{E}}_0\mathbf{c}_0\right\|_\infty\le\|\mathbf{e}_0\|_\infty^L+\left(\max_{1\le i\le d}\sum_{j=1}^{L+1}\left|\left[\widehat{\mathbf{E}}_0\right]_{i,j}\right|\right)\|\mathbf{c}_0\|_\infty\le\|\mathbf{e}_0\|_\infty^L+2\delta<\frac{1}{2}.$$

Now, we assume $\|\mathbf{e}_k\|_\infty<\frac{1}{2}$ holds. Then,

$$\|\mathbf{e}_{k+1}\|_\infty\le\|\mathbf{e}_k\|_\infty^L+\left\|\widehat{\mathbf{E}}_k\mathbf{c}_k\right\|_\infty\le\|\mathbf{e}_k\|_\infty^L+\left(\max_{1\le i\le d}\sum_{j=1}^{L+1}\left|\left[\widehat{\mathbf{E}}_k\right]_{i,j}\right|\right)\|\mathbf{c}_k\|_\infty\le\|\mathbf{e}_k\|_\infty^L+2\delta<\frac{1}{2}.$$

Since $\delta<\frac{1}{8}$, we have $2\delta<\frac{1}{4}$. By Lemma 3, we can conclude that:

$$\|\mathbf{e}_k\|_\infty\le e_0^{L^k}+(1+\epsilon)2\delta,\quad\text{where}\quad\frac{1}{16}>\epsilon\to 0\quad\text{as}\quad k\to\infty.$$

We finish the proof. $\square$

**Proposition 5** (Upper Bound First-Order Derivative)**.** *Suppose that* $\left|C_{\{k,l\}} - 1\right| = \delta < \frac{1}{8}$, $\forall\,(k, l+1) \in [K] \times [L]$ *and* $\|\mathbf{a}\|_\infty \leq 1$*, then we have:*

$$\frac{\partial \ell(\mathbf{C}, \mathbf{A})}{\partial C_{\{k,l\}}} \leq 4\left(\frac{1}{3}\right)^l \bar{b}_v^2 d.$$

*Proof.* Note that:

$$\frac{\partial \ell(\mathbf{C}, \mathbf{A})}{\partial C_{\{k,l\}}} = \sum_{\hat{k}=k}^{K} \left\langle \frac{\partial \ell(\mathbf{C}, \mathbf{A})}{\partial \mathbf{e}_{\hat{k}}}, \frac{\partial \mathbf{e}_{\hat{k}}}{\partial \mathbf{e}_{\hat{k}-1}} \circ \cdots \circ \frac{\partial \mathbf{e}_{k+1}}{\partial C_{\{k,l\}}} \right\rangle.$$

By Eq. (22), we also have:

$$\frac{\partial \mathbf{e}_{k+1}}{\partial \mathbf{e}_k} = \mathrm{Diag}\left(L\mathbf{e}_k^{L-1} + \widehat{\mathbf{E}}'_k \mathbf{c}_k\right),$$

where $\mathrm{Diag}(\mathbf{e})$ is a diagonal matrix with the diagonal entries being $\mathbf{e}$, and:

$$\widehat{\mathbf{E}}'_k \in \mathbb{R}^{d \times (L+1)} := \left[\mathbf{0}; \mathbf{e}_k^0; \cdots; L\mathbf{e}_k^{L-1}\right];$$

here $\mathbf{0} \in \mathbb{R}^d$ is the all zero vector. Hence, it holds that:

$$\left\|\frac{\partial \mathbf{e}_{k+1}}{\partial \mathbf{e}_k}\right\| = \left\|L\mathbf{e}_k^{L-1} + \widehat{\mathbf{E}}'_k \mathbf{c}_k\right\|_\infty \leq Le_k^{L-1} + \frac{e_k}{(1-e_k)^2} \cdot \delta, \tag{26}$$

where $e_k = \|\mathbf{e}_k\|_\infty$. By Proposition 4, we know that $e_k \leq e_0^{L^k} + \frac{17}{8}\delta < \frac{1}{3}$. Thus, we have:

$$\left\|\frac{\partial \mathbf{e}_{k+1}}{\partial \mathbf{e}_k}\right\| \leq \frac{1}{4}. \tag{27}$$

It is easy to see:

$$\frac{\partial \ell(\mathbf{C}, \mathbf{A})}{\partial \mathbf{e}_k} = \sum_{\hat{k}=k}^{K} \left\langle \frac{\partial \ell(\mathbf{C}, \mathbf{A})}{\partial \mathbf{e}_{\hat{k}}}, \frac{\partial \mathbf{e}_{\hat{k}}}{\partial \mathbf{e}_{\hat{k}-1}} \circ \cdots \circ \frac{\partial \mathbf{e}_{k+1}}{\partial \mathbf{e}_k} \right\rangle.$$

Hence, we have

$$\left\|\frac{\partial \ell(\mathbf{C}, \mathbf{A})}{\partial \mathbf{e}_{\hat{k}}}\right\|_2 \leq \sum_{\hat{k}=k}^{K} \left(\left\|\mathbf{v}_{\hat{k}} \odot \mathbf{a} \odot \left(\mathbf{v}_{\hat{k}} \odot \mathbf{a} \odot \mathbf{e}_{\hat{k}}\right)\right\|_2\right) \cdot 4^{k-\hat{k}} < 2\bar{b}_v^2\sqrt{d}.$$

According to Eq. (22), we notice:

$$\left\|\frac{\partial \mathbf{e}_{k+1}}{\partial C_{\{k,l\}}}\right\|_2 = \left\|\mathbf{e}_k^{l+1} - \mathbf{e}_k^l\right\|_2 \leq 2\left\|\mathbf{e}_k^l\right\|_2.$$

Combing all things together, we conclude that:

$$\left|\frac{\partial \ell(\mathbf{C}, \mathbf{A})}{\partial C_{\{k,l\}}}\right| \leq \sum_{\hat{k}=k}^{K} 4^{k-\hat{k}} \left(4\bar{b}_v^2\sqrt{d}\left\|\mathbf{e}_k^l\right\|_2\right) \leq 4\left(\frac{1}{3}\right)^l \bar{b}_v^2 d.$$

We finish the proof. $\square$

**Proposition 6** (Second Order Approximation)**.** *Define a set of coefficient collection around* $\mathbf{1}$*:*

$$\mathcal{C}^* := \left\{\mathbf{C} \mid \forall\, \mathbf{C}_{\{k,l\}} \in \mathbf{C},\ \left|C_{\{k,l\}} - 1\right| \leq \frac{1}{8}\right\}, \quad (k,l) \in [K] \times [L], \quad L \geq 4.$$

*For any* $\mathbf{C}_1, \mathbf{C}_2 \in \mathcal{C}^*$*, we have:*

$$\ell(\mathbf{C}_1, \mathbf{A}) = \ell(\mathbf{C}_2, \mathbf{A}) + \left\langle \frac{\partial \ell(\mathbf{C}_2, \mathbf{A})}{\partial \mathbf{C}}, \mathbf{C}_1 - \mathbf{C}_2 \right\rangle + \mathcal{O}\left(\bar{b}_v^2 dKL\right)\|\mathbf{C}_1 - \mathbf{C}_2\|_2^2.$$

*Proof.* Note that for any $\mathbf{C} \in \mathcal{C}^*$, we have:

$$\frac{\partial^2 \ell(\mathbf{C},\mathbf{A})}{\partial C_{\{k,l\}}\partial C_{\{k',l'\}}} = \left\langle \frac{\partial \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K}, \frac{\partial^2 \mathbf{e}_K}{\partial C_{\{k,l\}}\partial C_{\{k',l'\}}} \right\rangle + \left\langle \frac{\partial \mathbf{e}_K}{\partial C_{\{k,l\}}}, \frac{\partial^2 \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K \partial \mathbf{e}_K} \frac{\partial \mathbf{e}_K}{\partial C_{\{k',l'\}}} \right\rangle$$
$$= \left\langle \frac{\partial \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K}, \frac{\partial \mathbf{e}_K}{\partial \mathbf{e}_{K-1}} \circ \frac{\partial^2 \mathbf{e}_{K-1}}{\partial C_{\{k,l\}}\partial C_{\{k',l'\}}} \right\rangle + \left\langle \frac{\partial \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K}, \frac{\partial^2 \mathbf{e}_K}{\partial \mathbf{e}_{K-1}\partial \mathbf{e}_{K-1}} \left( \frac{\partial \mathbf{e}_{K-1}}{\partial C_{\{k,l\}}}, \frac{\partial \mathbf{e}_{K-1}}{\partial C_{\{k',l'\}}} \right) \right\rangle$$
$$+ \left\langle \frac{\partial \mathbf{e}_K}{\partial C_{\{k,l\}}}, \frac{\partial^2 \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K \partial \mathbf{e}_K} \frac{\partial \mathbf{e}_K}{\partial C_{\{k',l'\}}} \right\rangle$$
$$= \sum_{\hat{k}=\max\{k,k'\}}^{K} \left\langle \frac{\partial \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K}, \frac{\partial \mathbf{e}_K}{\partial \mathbf{e}_{K-1}} \circ \cdots \circ \frac{\partial^2 \mathbf{e}_{\hat{k}+1}}{\partial \mathbf{e}_{\hat{k}} \partial \mathbf{e}_{\hat{k}}} \left( \frac{\partial \mathbf{e}_{\hat{k}}}{\partial C_{\{k,l\}}}, \frac{\partial \mathbf{e}_{\hat{k}}}{\partial C_{\{k',l'\}}} \right) \right\rangle + \mathbf{\Delta}(k,k',l,l'),$$

where we let $\mathbf{e}_{k+1} \coloneqq \ell(\mathbf{C},\mathbf{A})$ with a little abuse of notation, and:

$$\mathbf{\Delta}(k,k',l,l') = \begin{cases} 0 & \text{if} \quad (k,l) = (k',l'), \\ \left\langle \frac{\partial \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K}, \frac{\partial \mathbf{e}_K}{\partial \mathbf{e}_{K-1}} \circ \cdots \circ \frac{\partial \left(\mathbf{e}_k^{l+1} - \mathbf{e}_k^l\right)}{\partial \mathbf{C}_{\{k',l'\}}} \right\rangle & \text{otherwise.} \end{cases}$$

Hence we can get the upper bound for the entry $\frac{\partial^2 \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{C}_{\{k,l\}}\partial \mathbf{C}_{\{k',l'\}}}$. From the proof in Proposition 5, we know that:

$$\left\| \frac{\partial \mathbf{e}_{k+1}}{\partial \mathbf{e}_k} \right\| \le \frac{1}{4}, \quad \left\| \frac{\partial \mathbf{e}_{\hat{k}}}{\partial C_{\{k,l\}}} \right\|_1 \le d \left(\frac{1}{4}\right)^{\hat{k}-k+l}, \quad \text{and} \quad \left\| \frac{\partial \mathbf{e}_{\hat{k}}}{\partial C_{\{k,l\}}} \right\|_2 \le 2\sqrt{d} \left(\frac{1}{4}\right)^{\hat{k}-k+l}.$$

Note that:

$$\frac{\partial^2 \mathbf{e}_{k+1}}{\partial \mathbf{e}_k \partial \mathbf{e}_k} = \mathrm{TDiag}\left( L(L-1)\mathbf{e}_k^{L-2} + \widehat{\mathbf{E}}_k'' \mathbf{c}_k \right),$$

where $\mathrm{TDiag}(\mathbf{e})$ is a three-dimensional diagonal tensor with the diagonal entries being $\mathbf{e}$, and:

$$\widehat{\mathbf{E}}_k'' \in \mathbb{R}^{d \times (L+1)} \coloneqq \left[ \mathbf{0}; \mathbf{0}; \mathbf{e}_k^0; \cdots; L(L-1)\mathbf{e}_k^{L-2} \right].$$

Hence, similar to Eq. (26), it holds that:

$$\left\| \frac{\partial^2 \mathbf{e}_{k+1}}{\partial \mathbf{e}_k \partial \mathbf{e}_k} \right\| = \left\| L(L-1)\mathbf{e}_k^{L-2} + \widehat{\mathbf{E}}_k' \mathbf{c}_k \right\|_\infty \le L(L-1)e_k^{L-2} + \frac{1 + 2e_k - 2e_k^2}{(1-e_k)^4} \cdot \delta < 3,$$

where $e_k = \|\mathbf{e}_k\|_\infty < \frac{1}{3}$ by Proposition 4. Moreover, we also note that:

$$\left\| \frac{\partial^2 \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_K \partial \mathbf{e}_K} \right\| = \left\| \mathrm{Diag}\left(\mathbf{v} \odot \mathbf{a}\right) \right\|^2 \le \bar{b}_v^2.$$

We also have:

$$\left\| \frac{\partial \ell(\mathbf{C},\mathbf{A})}{\partial \mathbf{e}_{\hat{k}}} \right\|_\infty \le \sum_{\hat{k}=k}^{K} \left( \left\| \mathbf{v}_{\hat{k}} \odot \mathbf{a} \odot \left( \mathbf{v}_{\hat{k}} \odot \mathbf{a} \odot \mathbf{e}_{\hat{k}} \right) \right\|_\infty \right) \cdot 4^{k-\hat{k}} < 2\bar{b}_v^2.$$

Hence, we have

$$\left| \frac{\partial^2 \ell(\mathbf{C},\mathbf{A})}{\partial C_{\{k,l\}}\partial C_{\{k',l'\}}} \right| \le 4\bar{b}_v^2 d \left(\frac{1}{4}\right)^{2K-\left(k+k'\right)+l+l'} + 3 \cdot 2\bar{b}_v^2 4d \left(\frac{1}{4}\right)^{2K-\left(k+k'\right)+l+l'-1} + \cdots$$
$$= 4\bar{b}_v^2 d \left(\frac{1}{4}\right)^{2K-\left(k+k'\right)+l+l'} + 24\bar{b}_v^2 d \sum_{\hat{k}=1}^{K-\max\{k,k'\}} \left(\frac{1}{4}\right)^{2K-\left(k+k'\right)+l+l'-\hat{k}}$$
$$< 29\bar{b}_v^2 d,$$

where we omit the term $\mathbf{\Delta}(k,k',l,l')$ since it is much smaller than the other sum term. Thus, we can conclude that

$$\left\| \frac{\partial^2 \ell(\mathbf{C},\mathbf{A},\mathbf{V})}{\partial \mathbf{C} \partial \mathbf{C}} \right\|_F = \left\| \left( \frac{\partial^2 \ell(\mathbf{C},\mathbf{A},\mathbf{V})}{\partial C_{\{k,l\}}\partial C_{\{k',l'\}}} \right)_{(k,k',l,l')} \right\|_F \le 29\bar{b}_v^2 KLd. \tag{28}$$

Combing all things together, we can see:

$$\ell(\mathbf{C}_1, \mathbf{A}) = \ell(\mathbf{C}_2, \mathbf{A}) + \left\langle \frac{\partial \ell(\mathbf{C}_2, \mathbf{A})}{\partial \mathbf{C}}, \mathbf{C}_1 - \mathbf{C}_2 \right\rangle + \left\langle \mathbf{C}_1 - \mathbf{C}_2, \frac{\partial^2 \ell(\mathbf{C}_\delta, \mathbf{A})}{\partial \mathbf{C} \partial \mathbf{C}} (\mathbf{C}_1 - \mathbf{C}_2) \right\rangle,$$

where the first equality comes from the second-order Taylor expansion, $\mathbf{C}_\delta := \mathbf{C}_1 + \delta(\mathbf{C}_2 - \mathbf{C}_1)$ and $\delta \in (0, 1)$ depends on $\mathbf{C}_1$ and $\mathbf{C}_2$. Due to Eq. (28), we conclude:

$$\ell(\mathbf{C}_1, \mathbf{A}) = \ell(\mathbf{C}_2, \mathbf{A}) + \left\langle \frac{\partial \ell(\mathbf{C}_2, \mathbf{A})}{\partial \mathbf{C}}, \mathbf{C}_1 - \mathbf{C}_2 \right\rangle + \mathcal{O}\big(\bar{b}_v^2 dKL\big) \|\mathbf{C}_1 - \mathbf{C}_2\|_2^2.$$

We finish the proof. □

## E.2 Proof of Theorem 1: Convergence Rate

*Proof.* Let us assume that $\mathbf{C}^{(t)} \in \mathcal{C}^*$ for all $t \in [T]$, where

$$\mathcal{C}^* := \left\{ \mathbf{C} \mid \forall\, C_{\{k,l\}} \in \mathbf{C},\ \left|\mathbf{C}_{\{k,l\}} - 1\right| \le \frac{1}{8} \right\}, \quad (k, l) \in [K] \times [L], \quad L \ge 4.$$

We will verity the boundness at the end.

We first provide the lower bound for the gradients.

**Claim 1.** *Under the same setting of this theorem, given:*

$$\mathcal{L}_n(\mathbf{C}) := \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{K} \left\| \mathbf{V}_{\{k,i\}} \big( \mathbf{A}_i \mathbf{X}_{\{k,i\}} \mathbf{A}_i - \mathbf{A}_i \big) \right\|_F^2.$$

*Suppose* $K = \Omega\big(nd\bar{b}_v^2 \bar{b}_a^2\big)$*, then with probability at least* $1 - \exp\left(-\mathcal{O}\big(K/\big(\bar{b}_v^2 \bar{b}_a^2\big)\big)\right)$*, We have:*

$$\left\| \mathcal{D}_{[\mathbf{C}=\mathbf{C}^{(t)}]} \mathcal{L}_n \right\|_2^2 \ge \frac{2\bar{b}_v^2 \bar{b}_a^2}{3} \mathcal{L}_n(\mathbf{C}).$$

*Proof.* First of all, we let

$$\boldsymbol{\upsilon}_{\{k,i\}} := \mathbf{v}_{\{k,i\}} \odot \mathbf{a}_i \odot \mathbf{e}_{\{k,i\}}^{(t)},$$

where $\mathbf{e}_{\{k,i\}}^{(t)}$ is the $k$-th layer error term of learnable D-Minv with the equivalent form Eq. (20), the subscript $\{k, i\}$ indicates the input is $\mathbf{A}_i$ and $k$-th layer, and $\mathbf{a}_i = \sigma_+(\mathbf{A}_i)$. We already have:

$$\mathcal{D}_{[\mathbf{C}=\mathbf{C}^{(t)}]} \mathcal{L}_n = \sum_{i=1}^{n} \sum_{k=1}^{K} \left\langle \boldsymbol{\upsilon}_{\{k,i\}},\ \frac{\partial\big(\boldsymbol{\upsilon}_{\{k,i\}}\big)}{\partial \mathbf{C}} \right\rangle.$$

**Derivative for the last layer:** We now consider the derivative for $C_{\{K,0\}}$, the first entry of $\mathcal{D}_{[\mathbf{C}=\mathbf{C}^{(t)}]}\mathcal{L}_n$. According to Eq. (22), we notice:

$$\frac{\partial \mathbf{e}_{k+1}}{\partial C_{\{k,l\}}} = \mathbf{e}_k^{l+1} - \mathbf{e}_k^l.$$

Hence, we have

$$\frac{\partial\big(\boldsymbol{\upsilon}_{\{K,i\}}\big)}{\partial C_{\{K,0\}}} = \mathbf{v}_{\{K,i\}} \odot \mathbf{a}_i \odot \left( \mathbf{e}_{\{K-1,i\}}^{(t)} - \mathbf{1} \right) \in \mathbb{R}^d.$$

Then, we can get

$$\mathcal{D}_{[C_{\{K,0\}} = C_{\{K,0\}}^{(t)}]} \mathcal{L}_n = \sum_{i=1}^{n} \left\langle \boldsymbol{\upsilon}_{\{K,i\}},\ \frac{\partial\big(\boldsymbol{\upsilon}_{\{K,i\}}\big)}{\partial C_{\{K,0\}}} \right\rangle = \sum_{i=1}^{n} \big(\mathbf{v}_{\{K,i\}}\big)^\top \left( \boldsymbol{\upsilon}_{\{K,i\}} \odot \mathbf{a}_i \odot \left( \mathbf{e}_{\{K-1,i\}}^{(t)} - \mathbf{1} \right) \right),$$

where $\boldsymbol{\upsilon}_{\{K,i\}} = \mathbf{v}_{\{K,i\}} \odot \mathbf{a}_i \odot \mathbf{e}_{\{K,i\}}^{(t)}$. Note that each entry of $\mathbf{v}_{\{j,i\}}$ obeys mean-zero bounded distribution; hence, it is a mean-zero $\bar{b}_v^2$-sub-Gaussian variable ($\mathrm{subG}(0, \bar{b}_v^2)$). Then, we have

$$\sum_{i=1}^{n} \big(\mathbf{v}_{\{K,i\}}\big)^\top \left( \boldsymbol{\upsilon}_{\{K,i\}} \odot \mathbf{a}_i \odot \left( \mathbf{e}_{\{K-1,i\}}^{(t)} - \mathbf{1} \right) \right) \sim \mathrm{subG}\left(0, \sigma_K^2\right), \quad \text{where} \quad \sigma_K^2 > \frac{4\bar{b}_v^2 \bar{b}_a^2}{9} \left( \sum_{i=1}^{n} \left\| \boldsymbol{\upsilon}_{\{K,i\}} \right\|_2^2 \right),$$

where the lower bound comes from the assumption $\|\mathbf{a}_i\|_\infty > \bar{b}_a$, and $\left\|\mathbf{e}^{(t)}_{\{K-1,i\}} - \mathbf{1}\right\|_\infty > \frac{2}{3}$ (by Propostion 4). Hence, we have:

$$\mathcal{D}_{[C_{\{K,0\}}=C^{(t)}_{\{K,0\}}]}\mathcal{L}_n \sim \text{subG}\left(0, \sigma_K^2\right), \quad \text{where} \quad \sigma_K^2 > \frac{4\bar{b}_v^2\bar{b}_a^2}{9}\left(\sum_{i=1}^n \left\|\boldsymbol{v}_{\{K,i\}}\right\|_2^2\right),$$

and

$$\left(\mathcal{D}_{[C_{\{K,0\}}=C^{(t)}_{\{K,0\}}]}\mathcal{L}_n\right)^2 \sim \text{subE}\left(\sigma_K^2, (\sigma_E)_K\right), \quad \text{where} \quad (\sigma_E)_K \geq \sigma_K^2,$$

here $\text{subE}(a, b)$ represents a sub-exponential variable with mean being $a$ and sub-exponential norm being $b$. The above conclusion comes from the fact "sub-exponential is sub-gaussian squared" (see Lemma 5.14 in Vershynin (2010)).

**Derivative for the other layer:** Now, we consider the derivative for $C_{\{\hat{k},0\}}$ for $\hat{k} < K$, Note that:

$$\mathcal{D}_{[C_{\{\hat{k},0\}}=C^{(t)}_{\{\hat{k},0\}}]}\mathcal{L}_n = \sum_{i=1}^n\sum_{k=\hat{k}}^K \left\langle \boldsymbol{v}_{\{k,i\}}, \frac{\partial\big(\boldsymbol{v}_{\{k,i\}}\big)}{\partial C_{\{\hat{k},0\}}}\right\rangle = \sum_{i=1}^n\sum_{k=\hat{k}}^K \left\langle \boldsymbol{v}_{\{k,i\}}, \frac{\partial\big(\boldsymbol{v}_{\{k,i\}}\big)}{\partial \mathbf{e}_{\{k-1,i\}}} \circ \cdots \circ \frac{\partial\Big(\mathbf{e}_{\{\hat{k},i\}}\Big)}{\partial C_{\{\hat{k},0\}}}\right\rangle.$$

We note that:

$$\left\|\frac{\partial\big(\boldsymbol{v}_{\{j,i\}}\big)}{\partial \mathbf{e}_{\{j-1,i\}}}\right\| = \left\|\text{Diag}\left(\mathbf{v}_{\{j,i\}} \odot \mathbf{a}_i\right)\right\| < \bar{b}_v, \quad \text{and} \quad \left\|\frac{\partial \mathbf{e}_{j+1}}{\partial \mathbf{e}_j}\right\| \leq \frac{1}{4}.$$

The shrink property comes from Proposition 5. Due to the exponential decay term $\left\|\frac{\partial \mathbf{e}_{j+1}}{\partial \mathbf{e}_j}\right\|$, the leading term of the derivative for $C_{\{\hat{k},0\}}$ is

$$\sum_{i=1}^n \left\langle \boldsymbol{v}_{\{\hat{k},i\}}, \frac{\partial\Big(\boldsymbol{v}_{\{\hat{k},i\}}\Big)}{\partial C_{\{\hat{k},0\}}}\right\rangle,$$

where we omit the term for all $k > \hat{k}$. Hence, similar to the case in the last layer, we have:

$$\mathcal{D}_{[C_{\{\hat{k},0\}}=C^{(t)}_{\{\hat{k},0\}}]}\mathcal{L}_n \sim \text{subG}\left(0, \sigma_{\hat{k}}^2\right), \quad \text{where} \quad \sigma_{\hat{k}}^2 > \frac{4\bar{b}_v^2\bar{b}_a^2}{9}\left(\sum_{i=1}^n \left\|\boldsymbol{v}_{\{\hat{k},i\}}\right\|_2^2\right),$$

and

$$\left(\mathcal{D}_{[C_{\{\hat{k},0\}}=C^{(t)}_{\{\hat{k},0\}}]}\mathcal{L}_n\right)^2 \sim \text{subE}\left(\sigma_{\hat{k}}^2, (\sigma_E)_{\hat{k}}\right), \quad \text{where} \quad (\sigma_E)_{\hat{k}} \geq \sigma_{\hat{k}}^2.$$

**Sum the square of derivative:** Note that, we have:

$$\left\|\mathcal{D}_{[\mathbf{C}=\mathbf{C}^{(t)}]}\mathcal{L}_n\right\|_2^2 \geq \sum_{k=1}^K \left(\mathcal{D}_{[C_{\{k,0\}}=C^{(t)}_{\{k,0\}}]}\mathcal{L}_n\right)^2.$$

Then by Bernstein-type inequality for the sub-exponential random variable, with probability at least $1 - \exp\left(-\mathcal{O}\big(K/\big(\bar{b}_v^2\bar{b}_a^2\big)\big)\right)$, it holds that

$$\sum_{j=1}^K \left(\mathcal{D}_{[C_{\{j,0\}}=C^{(t)}_{\{j,0\}}]}\mathcal{L}_n\right)^2 > \frac{\bar{b}_v^2\bar{b}_a^2}{3}\left(\sum_{i=1}^n\sum_{k=1}^K \left\|\boldsymbol{v}_{\{k,i\}}\right\|_2^2\right) = \frac{2\bar{b}_v^2\bar{b}_a^2}{3}\mathcal{L}_n(\mathbf{C}).$$

Till now we have completed the proof for one particular vector collection $\{\boldsymbol{v}_{\{k,i\}}\}$. In general, when $\big(K/\big(\bar{b}_v^2\bar{b}_a^2\big)\big)$ is large enough, we can prove that the above inequality holds for arbitrary $\{\boldsymbol{v}_{\{k,i\}}\}$ with high probability. Taking $\epsilon$-net over all possible vectors $\{\boldsymbol{v}_{\{k,i\}}\}$, then applying union bound, the above inequality holds with probability at least:

$$1 - \exp\left(-\mathcal{O}\big(K/\big(\bar{b}_v^2\bar{b}_a^2\big) - d\ln(d)\big)\right),$$

where the $\epsilon$-net has nothing to do with $K$ and $n$ since any $\boldsymbol{v}_{\{k,i\}}$ is contained in a $\ell_\infty$ ball with the radius less than $\bar{b}_v/3$. Due to the assumption that $K = \Omega\big(d\bar{b}_v^2\bar{b}_a^2\big)$, we can conclude that the desired result holds for all choices of $\{\boldsymbol{v}_{\{k,i\}}\}$. Now, we finish the proof. $\square$

We continue the proof for the convergence. Recall that

$$\mathcal{L}_n(\mathbf{C}) := \frac{1}{2nK}\sum_{i=1}^{n}\sum_{k=1}^{K}\left\|\mathbf{V}_{\{k,i\}}\big(\mathbf{A}_i\mathbf{X}_{\{k,i\}}\mathbf{A}_i - \mathbf{A}_i\big)\right\|_F^2,$$

By Proposition 6, we have:

$$\mathcal{L}_n(\mathbf{C}^{(t+1)}) = \mathcal{L}_n(\mathbf{C}^{(t)}) + \left\langle \frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}, \mathbf{C}^{(t+1)} - \mathbf{C}^{(t)} \right\rangle + \mathcal{O}\big(\bar{b}_v^2 dL\big)\left\|\mathbf{C}^{(t+1)} - \mathbf{C}^{(t)}\right\|_2^2,$$

where the order of $K$ is 0 than 1 since we have a factor $\frac{1}{K}$ compared to the function in Proposition 6. Since we adopt the GD method, we have

$$\mathbf{C}^{(t)} - \mathbf{C}^{(t+1)} = \eta\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}},$$

i.e.,

$$\mathcal{L}_n(\mathbf{C}^{(t+1)}) = \mathcal{L}_n(\mathbf{C}^{(t)}) - \eta\left\|\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}\right\|_2^2 + \eta^2\mathcal{O}\big(\bar{b}_v^2 dL\big)\left\|\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}\right\|_2^2. \tag{29}$$

By Claim 1, when $K = \Omega\big(d\bar{b}_v^2\bar{b}_a^2\big)$, with probability at least $1 - \exp\big(-\mathcal{O}\big(K/\big(\bar{b}_v^2\bar{b}_a^2\big)\big)\big)$, we already have:

$$\left\|\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}\right\|_2^2 \geq \frac{2\bar{b}_v^2\bar{b}_a^2}{3nK}\mathcal{L}_n(\mathbf{C}).$$

By our choice of $\eta = \Theta\big(1/\bar{b}_v^2 dL\big)$, we conclude

$$\mathcal{L}_n(\mathbf{C}^{(t+1)}) = \left(1 - \Omega\left(\frac{\eta\bar{b}_v^2\bar{b}_a^2}{nK}\right)\right)\mathcal{L}_n(\mathbf{C}^{(t)}).$$

Hence, in order to achieve accuracy $\epsilon$, we require

$$T = \frac{nK}{\bar{b}_v^2\bar{b}_a^2\eta}\ln\left(\frac{1}{\epsilon}\right)\mathcal{L}_n(\mathbf{C}^{(0)}) = \mathcal{O}\left(\frac{dLKn}{\bar{b}_a^2}\ln\left(\frac{1}{\epsilon}\right)\right) = \mathcal{O}\big(dLKn\bar{b}_a^{-2}\ln(1/\epsilon)\big),$$

where the first equality comes from our choice of $\eta$.

**Verify the boundness**: At last, we verify that $\mathbf{C}^{(t)} \in \mathcal{C}^*$ for all $t \in [T]$. Form Eq. (29), we have:

$$\eta\left\|\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}\right\|_2^2 \leq 2\Big(\mathcal{L}_n(\mathbf{C}^{(t)}) - \mathcal{L}_n(\mathbf{C}^{(t+1)})\Big),$$

then:

$$\begin{aligned}\left\|\mathbf{C}^{(T)} - \mathbf{C}^{(0)}\right\|_\infty &= \left\|\eta\sum_{t=1}^{T}\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}\right\|_\infty \leq \eta\left\|\sum_{t=1}^{T}\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}\right\|_2 \leq \eta\sum_{t=1}^{T}\left\|\frac{\partial \mathcal{L}_n(\mathbf{C}^{(t)})}{\partial \mathbf{C}}\right\|_2 \\ &\leq \sqrt{2\eta}\sum_{t=1}^{T}\sqrt{\mathcal{L}_n(\mathbf{C}^{(t)}) - \mathcal{L}_n(\mathbf{C}^{(t+1)})} \leq \sqrt{2\eta T\mathcal{L}_n(\mathbf{C}^{(1)})} \leq \sqrt{\frac{2nK}{\bar{b}_v^2\bar{b}_a^2}\ln\left(\frac{1}{\epsilon}\right)\mathcal{L}_n(\mathbf{C}^{(0)})\mathcal{L}_n(\mathbf{C}^{(1)})} \\ &= \mathcal{O}\left(\frac{d\sqrt{n}\ln(1/\epsilon)}{K^{2\alpha-1/2}}\right),\end{aligned}$$

where the last equality is from the fact $\left\|\mathbf{C}^{(0)} - \mathbf{1}\right\|_\infty < K^{-\alpha}$, which implies $\mathcal{L}_n(\mathbf{C}^{(0)}) = \mathcal{O}\big(\frac{1}{K^{2\alpha}}\big)$ according to Proposition 4. By setting large enough $K$, we can easily have

$$\left\|\mathbf{C}^{(T)} - \mathbf{C}^{(0)}\right\|_\infty \leq \frac{1}{16}.$$

Recall that our setting for $\mathbf{C}^{(0)}$ in Algorithms 1, we have $\left\|\mathbf{C}^{(0)} - \mathbf{1}\right\|_\infty < \frac{1}{16}$. Hence, we can conclude that

$$\left\|\mathbf{C}^{(T)} - \mathbf{1}\right\|_\infty \leq \left\|\mathbf{C}^{(T)} - \mathbf{C}^{(0)}\right\|_\infty + \left\|\mathbf{C}^{(0)} - \mathbf{1}\right\|_\infty \leq \frac{1}{8}.$$

We now can conclude $\mathbf{C}^{(t)} \in \mathcal{C}^*$ for all $t \in [T]$, and finish the whole proof. □

### E.3 Proof of Proposition 2: Local Lipschitz Smoothness

*Proof.* Note that $\mathbf{P}_{\mathbf{A}}\mathbf{E}_k = \widetilde{\mathbf{E}}_k$, where $\widetilde{\mathbf{E}}_k$ comes from Eq. (19). By Proposition 4, we know $\left\|\widetilde{\mathbf{E}}_k\right\| < \frac{1}{3}$, hence, by the "same norm upper bound" assumption, we have:

$$\left\|\mathbf{P}_{\mathbf{A}}\widetilde{\mathbf{E}}\right\| < \frac{1}{3}, \quad \text{and} \quad \left\|\mathbf{P}_{\mathbf{A}}\widehat{\mathbf{E}}\right\| < \frac{1}{3}.$$

By the mean value theorem, we have:

$$\mathcal{H}_k\left(\widehat{\mathbf{E}}\right) - \mathcal{H}_k\left(\widetilde{\mathbf{E}}\right) = \mathcal{D}_{[\mathbf{E}_k=\mathbf{E}_\delta]}\mathcal{H}_k\left(\widehat{\mathbf{E}} - \widetilde{\mathbf{E}}\right),$$

where $\mathbf{E}_\delta = \widetilde{\mathbf{E}} + \delta\left(\widetilde{\mathbf{E}} - \widehat{\mathbf{E}}\right)$ for some $\delta \in (0,1)$. Then

$$\left|\mathcal{H}_k\left(\widehat{\mathbf{E}}\right) - \mathcal{H}_k\left(\widetilde{\mathbf{E}}\right)\right| \le \left\|\mathcal{D}_{[\mathbf{E}_k=\mathbf{E}_\delta]}\mathcal{H}_k\right\| \cdot \left\|\left(\widehat{\mathbf{E}} - \widetilde{\mathbf{E}}\right)\right\|_*,$$

where $\|\cdot\|_*$ is the nuclear norm. Hence, the Lipschitz constant of $\mathcal{H}_k$ is bounded by $\left\|\mathcal{D}_{[\mathbf{E}_k=\mathbf{E}_\delta]}\mathcal{H}_k\right\|$. By the chain rule, we have:

$$\mathcal{D}_{[\mathbf{E}_k=\mathbf{E}_\delta]}\mathcal{H}_k = \sum_{\hat{k}=k}^{K}\left(\frac{\partial\mathcal{R}\left(\mathbf{G}_{\hat{k},\delta}\right)}{\partial\mathcal{G}_{\hat{k}-1}} \circ \frac{\partial\mathcal{G}_{\hat{k}-1}\left(\mathbf{G}_{\hat{k}-2,\delta}\right)}{\partial\mathcal{G}_{\hat{k}-2}} \circ \cdots \circ \frac{\partial\mathcal{G}_k(\mathbf{E}_\delta)}{\partial\mathbf{E}}\right),$$

where $\mathbf{G}_{\hat{k},\delta} := \mathcal{G}_{\hat{k}} \circ \cdots \circ \mathcal{G}_k(\mathbf{E}_\delta)$. Then, it holds that:

$$\left\|\mathcal{D}_{[\mathbf{E}_k=\mathbf{E}_\delta]}\mathcal{H}_k\right\| \le \sum_{\hat{k}=k}^{K}\left(\left\|\frac{\partial\mathcal{R}\left(\mathbf{G}_{\hat{k},\delta}\right)}{\partial\mathcal{G}_{\hat{k}-1}}\right\| \cdot \left\|\frac{\partial\mathcal{G}_{\hat{k}-1}\left(\mathbf{G}_{\hat{k}-2,\delta}\right)}{\partial\mathcal{G}_{\hat{k}-2}}\right\| \cdots \left\|\frac{\partial\mathcal{G}_k(\mathbf{E}_\delta)}{\partial\mathbf{E}}\right\|\right),$$

On one hand, for any $\hat{k} \ge k$, similar to Eq.(26) and Eq. (27), we have:

$$\left\|\frac{\partial\mathcal{G}_{\hat{k}+1}\left(\mathbf{G}_{\hat{k},\delta}\right)}{\partial\mathcal{G}_{\hat{k}}}\right\| \le Lg_{\hat{k}}^{L-1} + \frac{g_{\hat{k}}}{\left(1-g_{\hat{k}}\right)^2} \cdot \delta < \frac{1}{4},$$

Given $\|\mathbf{P}_{\mathbf{A}}\mathbf{E}_\delta\| < \frac{1}{3}$, it is easy to see that $\left\|\mathbf{G}_{\hat{k},\delta}\right\| < \frac{1}{3}$ for all $\hat{k} \ge k$ by Proposition 4, then:

$$\left\|\frac{\partial\mathcal{R}\left(\mathbf{G}_{\hat{k},\delta}\right)}{\partial\mathcal{G}_{\hat{k}-1}}\right\| = \left\|\mathbf{v}_{\hat{k}} \odot \mathbf{a} \odot \left(\mathbf{v}_{\hat{k}} \odot \mathbf{a} \odot \sigma_+\left(\mathbf{G}_{\hat{k},\delta}\right)\right)\right\|_\infty = \mathcal{O}(1).$$

where $g_{\hat{k}} = \left\|\mathbf{G}_{\hat{k},\delta}\right\|$. Combing all things together, we can conclude that:

$$\left\|\mathcal{D}_{[\mathbf{E}_k=\mathbf{E}_\delta]}\mathcal{H}_k\right\|_* \le C\sum_{\hat{k}=k}^{K}\left(\left(\frac{1}{4}\right)^{K-k}\right) \le 2,$$

where $C > 0$ is some proper constant. We finish the proof. □

### E.4 Proof of Theorem 2: Generalization

*Proof.* When let $\left\|\mathbf{C}^{(t)} - \mathbf{1}\right\| = \mathcal{O}(K^{-\alpha})$, we can easily conclude that $\mathcal{L}_n(\mathbf{C}^{(0)}) = \mathcal{O}\left(\frac{1}{K^{2\alpha-1}}\right)$ by Proposition 4. By the last part of Theorem 1's proof, we know that $\mathbf{C}^{(t)} \in \mathcal{C}_d^*$ for all $t \le T$, where:

$$\mathcal{C}_d^* := \left\{\mathbf{C} : \left\|\mathbf{C} - \mathbf{C}^{(0)}\right\|_F \le \widetilde{\mathcal{O}}\left(\frac{\sqrt{n}d}{K^{2\alpha-1/2}}\right)\right\}$$

**Part One**: We first find the upper bound for the empirical Rademacher complexity $\mathrm{Rad}_n(\mathcal{L})$ of our loss function class $\mathcal{L}$ (see Section G for definitions), where

$$\tilde{\mathcal{L}}_n := \left\{ \frac{1}{K}\ell(\mathbf{C}, \mathbf{A}) : \mathbf{C} \in \mathcal{C}_d^* \right\}.$$

See Eq. (25) for the definition of $\ell(\mathbf{C}, \mathbf{A})$. To bound the Rademacher complexity of $\mathcal{L}$, we invoke covering numbers by Lemma 5. First of all, we can easily verify the condition of Lemma 5 by Proposition 4 and Proposition 2. Due to Lemma 5, we have:

$$\ln \mathcal{N}\left( \tilde{\mathcal{L}}_n, \sum_{k=1}^{K} 2\epsilon_k, L_2(P_n, \|\cdot\|) \right) \leq \sum_{k=1}^{K} \ln \mathcal{N}(\mathcal{G}_k, \epsilon_k, L_2(P_n, \|\cdot\|)) + \ln \mathcal{N}(\mathcal{R}, \epsilon_k, L_2(P_n, \|\cdot\|)),$$

where we refer to Proposition 2 for the definition of $\mathcal{G}_k$ and $\mathcal{R}$. We first note that the term $\ln \mathcal{N}(\mathcal{R}, \epsilon_k, L_2(P_n, \|\cdot\|))$ is simply 0 because there is no learnable parameters for the output function $\mathcal{R}(\cdot)$. By our general analysis in Eq. (22), we have:

$$\ln \mathcal{N}(\mathcal{G}_k, \epsilon_k, L_2(P_n, \|\cdot\|)) = \ln \mathcal{N}\left( \left\{ \widehat{\mathbf{E}}_k \mathbf{c} : \mathbf{c} \in \mathbb{R}^L, \|\mathbf{c}\|_\infty \leq \frac{1}{4} \right\}, \epsilon_k, \|\cdot\|_F \right).$$

Note that $\left\| \widehat{\mathbf{E}}_k \right\|_F^2 = \mathcal{O}(d)$. Moreover, by proof in Claim 1, if $K^{2\alpha - 1/2} = \Omega(\sqrt{n})$, similar to the proof of the boundness verification, it holds that

$$\|\mathbf{c}\|_2^2 < \widetilde{\mathcal{O}}(d),$$

where $\widetilde{\mathcal{O}}(\cdot)$ hides log factors, and then according to Lemma 4, we have:

$$\ln \mathcal{N}(\mathcal{G}_k, \epsilon_k, L_2(P_n, \|\cdot\|)) \leq \left\lceil \frac{d^3}{\epsilon_k^2} \right\rceil \ln(2L),$$

By setting $\epsilon_k = \epsilon$, we have $\sum_{k=1}^{K} \frac{1}{K}\epsilon_k = \epsilon$, where the factor $\frac{1}{K}$ is from the fact Proposition 2. Then, according to Lemma 5, we have:

$$\ln \mathcal{N}\left( \tilde{\mathcal{L}}_n, \epsilon, L_2(P_n, \|\cdot\|) \right) \leq \widetilde{\mathcal{O}}\left( \frac{d^3 K}{\epsilon^2} \right)$$

where $\widetilde{\mathcal{O}}(\cdot)$ hides log factors. By Dudley's entropy theorem (Lemma 7), we can get:

$$\mathrm{Rad}_n(\tilde{\mathcal{L}}_n) = \widetilde{\mathcal{O}}\left( \sqrt{\frac{d^3 K}{n}} \right).$$

Thus, applying the standard Rademacher generalization bound (Lemma 6), with probability $1 - \delta$ over the training data, it holds that:

$$\mathop{\mathbb{E}}_{\mathbf{A} \sim \mathcal{P}_{\mathbf{A}}} \left[ \mathcal{L}_n\left( \mathbf{C}^{(t)} \right) \right] \leq \mathcal{L}_n\left( \mathbf{C}^{(t)} \right) + \widetilde{\mathcal{O}}\left( \sqrt{\frac{d^3 K}{n}} \right) + \mathcal{O}\left( \sqrt{\frac{\ln(1/\delta)}{n}} \right).$$

**Part Two**: We now ready to provide the generalization bound for the case $K \gg n$, w.o.l.g. we let $K = \Omega(n)$. When let $\left\| \mathbf{C}^{(t)} - \mathbf{1} \right\| = \mathcal{O}(K^{-\alpha})$, we can easily conclude that $\mathcal{L}_n(\mathbf{C}^{(0)}) = \mathcal{O}\left( \frac{1}{K^{2\alpha}} \right)$ by Proposition 4. Recall the last part of Theorem 1's proof (boundness verification), we have:

$$\left\| \mathbf{C}^{(t)} - \mathbf{C}^{(0)} \right\|_F = \widetilde{\mathcal{O}}\left( \frac{\sqrt{n} d}{K^{2\alpha - 1/2}} \right), \quad \forall t \in [T].$$

Hence, by Proposition 6, with $\tilde{\ell}(\cdot) := \frac{1}{K}\ell(\cdot)$, we have:

$$\begin{aligned}
\tilde{\ell}(\mathbf{C}^{(t)}) &= \tilde{\ell}(\mathbf{C}^{(0)}) + \left\langle \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)})}{\partial \mathbf{C}}, \mathbf{C}^{(t)} - \mathbf{C}^{(0)} \right\rangle + \mathcal{O}(\bar{b}_v^2 d L) \left\| \mathbf{C}^{(t)} - \mathbf{C}^{(0)} \right\|_2^2 \\
&= \tilde{\ell}(\mathbf{C}^{(0)}) + \left\langle \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)})}{\partial \mathbf{C}}, \mathbf{C}^{(t)} - \mathbf{C}^{(0)} \right\rangle + \widetilde{\mathcal{O}}\left( \frac{\bar{b}_v^2 d^3 n L}{K^{4\alpha - 1}} \right) \\
&= \tilde{\ell}(\mathbf{C}^{(0)}) + \left\langle \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)})}{\partial \mathbf{C}}, \mathbf{C}^{(t)} - \mathbf{C}^{(0)} \right\rangle + \widetilde{\mathcal{O}}\left( \frac{\bar{b}_v^2 d^3 L}{K^{\alpha/2}} \right),
\end{aligned}$$

where $\widetilde{\mathcal{O}}(\cdot)$ hides log factors, and the last equality is due to $K = \Omega(n)$ and $\alpha \geq 1$. Thus, it holds that:

$$\mathrm{Rad}_n(\tilde{\mathcal{L}}_n) \leq \mathrm{Rad}_n\left(\left\{\left\langle \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)})}{\partial \mathbf{C}}, \mathbf{C} - \mathbf{C}^{(0)} \right\rangle : \mathbf{C} \in \mathcal{C}_d^*\right\}\right) + \widetilde{\mathcal{O}}\left(\frac{\bar{b}_v^2 d^3 L}{K^{\alpha/2}}\right),$$

where $\tilde{\mathcal{L}}_n \coloneqq \left\{\tilde{\ell}(\mathbf{C}, \mathbf{A}) : \mathbf{C} \in \mathcal{C}_d^*\right\}$, and we omit the term $\tilde{\ell}(\mathbf{C}^{(0)})$ due to $\mathrm{Rad}_n\left(\tilde{\ell}(\mathbf{C}^{(0)})\right) = 0$. By Cauchy-Schwarz inequality, we have:

$$\begin{aligned}&\mathrm{Rad}_n\left(\left\{\left\langle \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)})}{\partial \mathbf{C}}, \mathbf{C} - \mathbf{C}^{(0)} \right\rangle : \mathbf{C} \in \mathcal{C}_d^*\right\}\right)\\ &\leq \widetilde{\mathcal{O}}\left(\frac{\sqrt{n}d}{K^{2\alpha-1/2}}\right) \mathop{\mathbb{E}}_{\alpha_i}\left[\left\|\frac{1}{n}\sum_{i=1}^n \alpha_i \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)}, \mathbf{A}_i)}{\partial \mathbf{C}}\right\|_F\right] \leq \widetilde{\mathcal{O}}\left(\frac{d}{\sqrt{n}K^{2\alpha-1/2}}\right)\sqrt{\mathop{\mathbb{E}}_{\alpha_i}\left[\left\|\sum_{i=1}^n \alpha_i \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)}, \mathbf{A}_i)}{\partial \mathbf{C}}\right\|_F^2\right]},\end{aligned}$$

where we apply Jensen's inequality to obtain the last inequality. Notice that:

$$\sqrt{\mathop{\mathbb{E}}_{\alpha_i}\left[\left\|\sum_{i=1}^n \alpha_i \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)}, \mathbf{A}_i)}{\partial \mathbf{C}}\right\|_F^2\right]} = \sqrt{\sum_{i=1}^n \left\|\frac{\partial \tilde{\ell}(\mathbf{C}^{(0)}, \mathbf{A}_i)}{\partial \mathbf{C}}\right\|_F^2}.$$

By Proposition 5, we have:

$$\left\|\frac{\partial \tilde{\ell}(\mathbf{C}^{(0)}, \mathbf{A}_i)}{\partial \mathbf{C}}\right\|_F^2 = \mathcal{O}\left(\bar{b}_v^4 d^2\right),$$

which implies:

$$\sqrt{\sum_{i=1}^n \left\|\frac{\partial \tilde{\ell}(\mathbf{C}^{(0)}, \mathbf{A}_i)}{\partial \mathbf{C}}\right\|_F^2} = \mathcal{O}\left(\sqrt{n}\bar{b}_v^2 d\right).$$

Hence, we have:

$$\mathrm{Rad}_n\left(\left\{\left\langle \frac{\partial \tilde{\ell}(\mathbf{C}^{(0)})}{\partial \mathbf{C}}, \mathbf{C} - \mathbf{C}^{(0)} \right\rangle : \mathbf{C} \in \mathcal{C}_d^*\right\}\right) = \widetilde{\mathcal{O}}\left(\frac{\bar{b}_v^2 d^2}{K^{2\alpha-1/2}}\right).$$

Combing all things together, we conclude:

$$\mathrm{Rad}_n(\tilde{\mathcal{L}}_n) = \widetilde{\mathcal{O}}\left(\frac{\bar{b}_v^2 d^3 L}{K^{\alpha/2}}\right).$$

Similar to the part one, we have:

$$\mathop{\mathbb{E}}_{\mathbf{A}\sim\mathcal{P}_{\mathbf{A}}}\left[\mathcal{L}_n\left(\mathbf{C}^{(t)}\right)\right] \leq \mathcal{L}_n\left(\mathbf{C}^{(t)}\right) + \widetilde{\mathcal{O}}\left(\frac{\bar{b}_v^2 d^3 L}{K^{\alpha/2}}\right) + \mathcal{O}\left(\sqrt{\frac{\ln(1/\delta)}{n}}\right).$$

We now finish the whole proof. □

## F Omitted Proof for D-SVD in Sections C

### F.1 Proof of Proposition 3

*Proof.* First of all, we provide the Lipschitz constant w.r.t. $\left(t_k, \mathbf{C}_k\right)$. Note that:

$$\mathbf{U}_{k+1} = \left(\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U\right)^{-1}(\mathbf{I} - \mathbf{E}_{K_{\mathrm{inv}}})\left(\mathbf{I} - \frac{t_U^*}{2}\mathbf{P}_U\right)\mathbf{U}_k \tag{30}$$

where

$$\mathbf{E}_{K_{\mathrm{inv}}} = \mathbf{I} - \left(\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U\right)\text{LD-Minv}_U\left(\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U, \mathbf{C}\right).$$

On one hand, by Eq. (26) in Proposition 5,we have:

$$\left\|\frac{\partial \mathbf{E}_{k+1}}{\partial \mathbf{E}_k}\right\| = \mathcal{O}\left(\frac{1}{K_{\text{inv}}}\right), \quad \forall k \in [K_{\text{inv}}]$$

On the other hand, we get:

$$\frac{\partial \mathbf{E}_{K_{\text{inv}}}}{\partial t_U^*} = \frac{\partial \mathbf{E}_{K_{\text{inv}}}}{\partial \mathbf{E}_{K_{\text{inv}}-1}} \frac{\partial \mathbf{E}_{K_{\text{inv}}-1}}{\partial \mathbf{E}_{K_{\text{inv}}-2}} \cdots \frac{\partial \mathbf{E}_1}{\partial t_U^*}.$$

When $K_{\text{inv}}$ is large enough, we can ignore $\mathbf{E}_{K_{\text{inv}}}$'s impact, and conclude that the Lipschitz constant of $f$ w.r.t. $\tilde{t}_k$ dominate by the other term in Eq. (30) rather than $\mathbf{E}_{K_{\text{inv}}}$. Recall that by Proposition 1, $\mathbf{E}_{K_{\text{inv}}}$ share the same row and the same column space with $\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U$. W.o.l.g. we can assume $\mathbf{E}_{K_{\text{inv}}}$ is diagonal, and by Proposition 4, we have $\mathbf{E}_{K_{\text{inv}}} = \mathcal{O}\big(1/\sqrt{K_{\text{inv}}}\big)$. Thus, for large $K_{\text{inv}}$, we have

$$\left\|\frac{\partial \mathbf{U}_{k+1}}{\partial \tilde{t}_{U_k}}\right\| = \left\|\frac{\partial \mathbf{U}_{k+1}}{\partial t_U^*}\right\| = \mathcal{O}\left(\left\|\frac{\partial\left(\left(\mathbf{I}+\frac{t_U^*}{2}\mathbf{P}_U\right)^{-1}\left(\mathbf{I}-\frac{t_U^*}{2}\mathbf{P}_U\right)\mathbf{U}_k\right)}{\partial t_U^*}\right\|\right)$$

From the spectral theorem, for a real skew-symmetric matrix the nonzero eigenvalues are all pure imaginary and thus are of the form $\lambda_1 i, -\lambda_1 i, \lambda_2 i, -\lambda_2 i, \ldots \lambda_1 i, -\lambda_1 i, \lambda_2 i, -\lambda_2 i, \ldots$ where each of the $\lambda_i$ are real. Hence, let $\mathbf{P}_U = \mathbf{U}_P \mathbf{\Lambda} \mathbf{U}_P^*$, where the entries of the diagonal of $\mathbf{\Lambda}$ are $\pm\lambda_i i$, and $\mathbf{U}_P$ is the unitary matrix, we have

$$\left(\mathbf{I}+\frac{t_U^*}{2}\mathbf{P}_U\right)^{-1}\left(\mathbf{I}-\frac{t_U^*}{2}\mathbf{P}_U\right)\mathbf{U}_k = \mathbf{U}_P\left(\left(\mathbf{I}+\frac{t_U^*}{2}\mathbf{\Lambda}\right)^{-1}\left(\mathbf{I}-\frac{t_U^*}{2}\mathbf{\Lambda}\right)\right)\mathbf{U}_P^*\mathbf{U}_k,$$

then

$$\frac{\partial\left(\mathbf{I}+\frac{t_U^*}{2}\mathbf{P}_U\right)^{-1}\left(\mathbf{I}-\frac{t_U^*}{2}\mathbf{P}_U\right)\mathbf{U}_k}{\partial t_U^*} = -\mathbf{U}_P\left(\mathbf{\Lambda}(\mathbf{I}+\frac{t_U^*}{2}\mathbf{\Lambda})^{-2}\right)\mathbf{U}_P^*\mathbf{U}_k.$$

Hence, we can easily obtain that:

$$\left\|\frac{\partial \mathbf{U}_{k+1}}{\partial t_U^*}\right\| = \mathcal{O}\left(\left\|\frac{\partial\left(\left(\mathbf{I}+\frac{t_U^*}{2}\mathbf{P}_U\right)^{-1}\left(\mathbf{I}-\frac{t_U^*}{2}\mathbf{P}_U\right)\mathbf{U}_k\right)}{\partial t_U^*}\right\|\right) = \mathcal{O}(\|\mathbf{\Lambda}\|) = \mathcal{O}(\|\mathbf{M}\|),$$

where the second and the third equations come from the facts:

$$\left|\frac{\lambda i}{\left(1+\frac{t_U^*}{2}\lambda i\right)^2}\right| < |\lambda| \quad \text{and} \quad |\lambda_1(\mathbf{P})| \leq \sigma_1(\mathbf{P}),$$

where $|\lambda_1(\cdot)|$ and $\sigma_1(\cdot)$ are the largest scale eigenvalue and singular value, respectively. Hence, we can conclude that the Lipschitz constant of $f$ w.r.t. $\tilde{t}_{U_k}$ is in the order of $\mathcal{O}(\|\mathbf{M}\|)$.

By the proof in Proposition 5, we have

$$\left\|\frac{\partial \mathbf{E}_{k+1}}{\partial C_{\{k,l\}}}\right\| = \left\|\mathbf{E}_k^{l+1} - \mathbf{E}_k^l\right\| \leq 2\left\|\mathbf{E}_k^l\right\|, \quad \forall k \in [K_{\text{inv}}].$$

Hence, we get:

$$\left\|\frac{\partial \mathbf{E}_{K_{\text{inv}}}}{\partial C_{\{k,l\}}}\right\| = \mathcal{O}\left({K_{\text{inv}}}^{(k-K_{\text{inv}})} 2\left\|\mathbf{E}_k^l\right\|\right) = \mathcal{O}\left({K_{\text{inv}}}^{(k-K_{\text{inv}}-l/2)}\right),$$

where the last equation is from $\|\mathbf{E}_k\| = \mathcal{O}\left(\frac{1}{K_{\text{inv}}}\right)$. Then, we can get:

$$\left\|\frac{\partial \mathbf{U}_{k+1}}{\partial C_{\{k,l\}}}\right\| \leq \left\|\frac{\partial \mathbf{E}_{K_{\text{inv}}}}{\partial C_{\{k,l\}}}\right\| = \mathcal{O}\left({K_{\text{inv}}}^{(k-K_{\text{inv}}-l/2)}\right),$$

where this equation is from the fact that $\mathbf{E}_{K_{\mathrm{inv}}}$ share the same row and the same column space with $\mathbf{I} \pm \frac{t_U^*}{2}\mathbf{P}_U$, and:

$$\left\| \left(\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U\right)^{-1} \left(\mathbf{I} - \frac{t_U^*}{2}\mathbf{P}_U\right) \right\| = 1.$$

Thus, we can easily verify that:

$$\sum_{k=1}^{K_{\mathrm{inv}}} \sum_{l=0}^{L} \left\| \frac{\partial \mathbf{U}_{k+1}}{\partial C_{\{k,l\}}} \right\| = \mathcal{O}(1).$$

Hence, we can conclude that the Lipschitz constant of $f$ w.r.t. $\mathbf{C}$ is in the order of $\mathcal{O}(1)$.

Because of the Lipschitz condition, we can have:

$$\ln \mathcal{N}(\mathcal{F}_k, \epsilon, \|\cdot\|) \leq \ln \mathcal{N}\left(\mathcal{C}_k, \frac{\epsilon}{\mathrm{Lip}(f)}, \ell_\infty\right),$$

where $\mathrm{Lip}(f)$ is the Lipschitz constant of $f$ w.r.t. the spectral norm $\|\cdot\|$. Note that:

$$\ln \mathcal{N}\left(\mathcal{C}_k, \frac{\epsilon}{\mathrm{Lip}(f)}, \ell_\infty\right) = \mathcal{O}\left(\ln \mathcal{N}\left(\{\tilde{t}_{U_k}\}, \frac{\epsilon}{\|\mathbf{M}\|}, \ell_\infty\right) + \ln \mathcal{N}\left(\{\mathbf{C}\}, \epsilon\sqrt{K_{\mathrm{inv}}}, \ell_\infty\right)\right).$$

Note the bounds on covering number:

$$\mathcal{N}\left(\{\mathbf{C}\}, \epsilon\sqrt{K_{\mathrm{inv}}}, \ell_\infty\right) \leq \left(1 + \frac{2}{\epsilon\sqrt{K_{\mathrm{inv}}}}\right)^{LK_{\mathrm{inv}}}.$$

Finally, we get:

$$\ln \mathcal{N}(\mathcal{F}_k, \epsilon, \|\cdot\|) = \mathcal{O}\left(LK_{\mathrm{inv}} \ln\left(\frac{1}{\epsilon\sqrt{K_{\mathrm{inv}}}}\right) + \ln\left(\frac{\|\mathbf{M}\|}{\epsilon}\right)\right).$$

We now finish the proof. □

## F.2 Proof of Theorem 3

*Proof.* Before adopting Lemma 5, we first verify the conditions of this Lemma. The boundness condition of the intermediate results is obvious since $\{\mathbf{U}_{k,i}\}$ and $\{\mathbf{V}_{k,i}\}$ are both from the Stiefel manifold.

On the other hand, we can easily verify the Lipschitz condition when $K_{\mathrm{inv}} \geq K_{\mathrm{svd}}$. Let $d = \max\{m, \hat{n}\}$. Recall the last part of Theorem 1's proof (boundness verification), we have:

$$\left\|\mathbf{C}^{(t)} - \mathbf{C}^{(0)}\right\|_F = \widetilde{\mathcal{O}}\left(\frac{\sqrt{n}d}{K_{\mathrm{inv}}}\right), \quad \forall t \in [T].$$

By the assumption $K = \Omega(nd)$, we have

$$\left\|\mathbf{C}^{(t)} - \mathbf{C}^{(0)}\right\|_F = \widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{K_{\mathrm{inv}}}}\right), \quad \forall t \in [T].$$

By Eq. (30) in the proof of Proposition 3, we have the Lipschitz constant of $f(\mathbf{U}_k \mid \mathbf{V}_k, \mathbf{M}, \tilde{t}_k, \mathbf{C}_k)$ w.r.t. $\mathbf{U}_k$ is bounded by:

$$\left\| \left(\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U\right)^{-1} (\mathbf{I} - \mathbf{E}_{K_{\mathrm{inv}}}) \left(\mathbf{I} - \frac{t_U^*}{2}\mathbf{P}_U\right) \right\| \leq 1 + \|\mathbf{E}_{K_{\mathrm{inv}}}\|.$$

where this inequality is from the fact that $\mathbf{E}_{K_{\mathrm{inv}}}$ share the same row and the same column space with $\mathbf{I} \pm \frac{t_U^*}{2}\mathbf{P}_U$, and:

$$\left\| \left(\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U\right)^{-1} \left(\mathbf{I} - \frac{t_U^*}{2}\mathbf{P}_U\right) \right\| = 1.$$

Due to $\left\|\mathbf{C}^{(t)} - \mathbf{C}^{(0)}\right\|_F = \widetilde{\mathcal{O}}\Big(\frac{1}{\sqrt{K_{\text{inv}}}}\Big)$, we have:

$$\left\|\Big(\mathbf{I} + \frac{t_U^*}{2}\mathbf{P}_U\Big)^{-1}(\mathbf{I} - \mathbf{E}_{K_{\text{inv}}})\Big(\mathbf{I} - \frac{t_U^*}{2}\mathbf{P}_U\Big)\right\| = \widetilde{\mathcal{O}}\Big(1 + \frac{1}{\sqrt{K_{\text{inv}}}}\Big).$$

Thus, the Lipschitz constant from any intermediate layer to output layer of D-SVD is bounded by:

$$\widetilde{\mathcal{O}}\left(\left(1 + \frac{1}{\sqrt{K_{\text{inv}}}}\right)^{2K_{\text{svd}}}\right) = \widetilde{\mathcal{O}}\left(\left(1 + \frac{1}{K_{\text{svd}}}\right)^{2K_{\text{svd}}}\right) = \widetilde{\mathcal{O}}(1).$$

where we use the assumption that $\sqrt{K_{\text{inv}}} = \Omega(K_{\text{svd}})$. We have verified the two conditions in Lemma 5.

By setting $\epsilon_k = \epsilon/K_{\text{svd}}$, we have $\sum_{k=1}^{K_{\text{svd}}} \epsilon_k = \epsilon$. Due to Lemma 5 and Proposition 3, we obtain:

$$\ln \mathcal{N}\Big(\tilde{\mathcal{M}}_n, \epsilon, L_2(P_n, \|\cdot\|)\Big) = \widetilde{\mathcal{O}}\left(K_{\text{svd}} L K_{\text{inv}} \ln\left(\frac{K_{\text{svd}}}{\sqrt{K_{\text{inv}}}\epsilon}\right)\right).$$

where $\widetilde{\mathcal{O}}(\cdot)$ hides some log factors.

By Dudley's entropy theorem (Lemma 7), we can get:

$$\text{Rad}_n(\tilde{\mathcal{M}}_n) = \widetilde{\mathcal{O}}\left(\sqrt{\frac{K_{\text{svd}}^3 L}{n}}\right),$$

where we utilize the fact that the integral of $\sqrt{\ln\Big(\frac{K_{\text{svd}}}{\epsilon\sqrt{K_{\text{inv}}}}\Big)}$ evaluated near zero is in the order of $\mathcal{O}\Big(\frac{K_{\text{svd}}}{\sqrt{K_{\text{inv}}}}\Big)$. Thus, applying the standard Rademacher generalization bound (Lemma 6), with probability $1 - \delta$ over the training data, it holds that:

$$\mathop{\mathbb{E}}_{\mathbf{M} \sim \mathcal{P}_{\mathbf{M}}}\left[\mathcal{M}_n\Big(\tilde{\mathbf{t}}^{(t)}\Big)\right] \leq \mathcal{M}_n\Big(\tilde{\mathbf{t}}^{(t)}\Big) + \widetilde{\mathcal{O}}\left(\sqrt{\frac{K_{\text{svd}}^3 L}{n}}\right) + \mathcal{O}\left(\sqrt{\frac{\ln(1/\delta)}{n}}\right).$$

We now finish the proof. □

## G AUXILIARY LEMMAS

**Lemma 3** (Bounded Iterates). *For $q \in \mathbb{N} \geq 4$, given $x_0, e \in \mathbb{R}$, such that $x_0 \leq \frac{1}{2}$ and $e \leq \frac{1}{4}$, let $x_k = x_{k-1}^q + e$, then:*

$$x_k \leq x_0^{q^k} + (1+\epsilon)e, \quad \forall k \geq 0, \quad \textit{where} \quad \epsilon \to 0 \quad \textit{as} \quad k \to \infty, \quad \textit{and} \quad \epsilon \leq \frac{1}{16}.$$

*Proof.* W.l.o.g, in this proof, we assume $x_0^{q^k} \ll e$, otherwise, the conclusion is obvious. We prove this lemma by induction. For $k = 0$, we have $x_0 = x_0^{q^0}$, the conclusion is obvious. Now, we assume the inequality holds for $x_k$, then:

$$x_{k+1} = x_k^q + e \leq \left(x_0^{q^k} + \frac{17e}{16}\right)^q + e \overset{(a)}{\leq} x_0^{q^{k+1}} + 2q(\frac{17e}{16})^{q-1} x_0^{q^k} + e,$$

where $(a)$ come from the Binomial theorem for $\Big(x_0^{q^k} + \frac{17e}{16}\Big)^q$ with $x_0^{q^k} \ll e$. It is easy to verify that:

$$0.8 \cdot \left(\frac{17e}{16}\right)^{q-2} < \frac{1}{16}, \quad \text{and} \quad 1 > 1.25 \cdot \frac{17}{8} q \cdot x_0^{q^k} \to 0 \quad \text{as} \quad k \to \infty,$$

hence

$$x_{k+1} \leq x_0^{q^{k+1}} + e + e\Big(4q(2e)^{q-2} x_0^{q^k}\Big) < x_0^{q^{k+1}} + (1+\epsilon)e.$$

By the induction, we finish the proof. □

Let $\mathcal{N}(U, \epsilon, |||\cdot|||)$ denote the least cardinality of any subset $V \subseteq U$ that covers $U$ at scale $\epsilon$ with norm $|||\cdot|||$, meaning:

$$\sup_{A \in U} \min_{B \in V} |||A - B||| \leq \epsilon.$$

Then we have the following lemma.

**Lemma 4** (Matrix Covering, Bartlett et al. (2017) Lemma 3.2)**.** *Let matrix $\mathbf{X} \in \mathbb{R}^{d \times L}$ be given with $\|\mathbf{X}\|_F \leq b$. Then*

$$\ln \mathcal{N}\Big(\Big\{\mathbf{XC} : \mathbf{C} \in \mathbb{R}^{L \times m}, \|\mathbf{C}\|_{2,1} \leq a\Big\}, \epsilon, \|\cdot\|_F\Big) \leq \left\lceil \frac{a^2 b^2}{\epsilon^2} \right\rceil \ln(2Lm),$$

*where $\|\mathbf{C}\|_{2,1} = \sum_{i=1}^m \|\mathbf{C}_{:,i}\|_2$.*

Let $P_n$ be a uniform distribution over $n$ points data $\mathcal{X}_n \coloneqq \{x_1, x_2, \cdots, x_n\}$. Then the $L_2(P_n, |||\cdot|||)$ norm of the function $f$ is defined as

$$\|f\|_{L_2(P_n, |||\cdot|||)} \coloneqq \left(\frac{1}{n} \sum_{i=1}^n |||f|||^2\right)^{\frac{1}{2}}.$$

Let $\mathcal{F}_1, \cdots, \mathcal{F}_k$ be a sequence of families of functions (which corresponds to families of single layer neural nets in the deep learning setting) and $\ell$ be a Lipschitz loss function. We study the family of compositions of $\ell$ and functions in $\mathcal{F}_i$'s:

$$\mathcal{L} \coloneqq \ell \circ \mathcal{F}_k \circ \mathcal{F}_{k-1} \cdots \circ \mathcal{F}_1 = \{\ell \circ f_k \circ f_{k-1} \circ \cdots \circ f_1 : \forall i, f_i \in \mathcal{F}_i\}.$$

**Lemma 5** (Abstraction of Techniques in Bartlett et al. (2017); Wei & Ma (2019))**.** *Assume:*

1. *Given the data set $\mathcal{X}_n$ and norm $|||\cdot|||$, for any $x \in \mathcal{X}$, $|||f_i \circ \cdots \circ f_1(x)||| \leq s_i$.*
2. *$\ell \circ f_k \circ \cdots \circ f_{i+1}$ is $\kappa_i$-Lipschitz for all $i$.*

*Then, we have the following covering number bound for $\mathcal{L}$ (for any choice $\epsilon_1, \ldots, \epsilon_k > 0$):*

$$\ln \mathcal{N}\left(\mathcal{L}, \sum_{i=1}^k \kappa_i \epsilon_i, L_2(P_n, |||\cdot|||)\right) \leq \sum_{i=1}^k \ln \mathcal{N}(\mathcal{F}_i, \epsilon_i, L_2(P_n, |||\cdot|||)).$$

The above lemma lemma says that if the intermediate variable (or the hidden layer) $f_i \circ \cdots \circ f_1(x)$ is bounded, and the composition of the rest of the functions $\ell \circ f_k \circ \cdots \circ f_{i+1}(x)$ is Lipschitz, then small covering number of local functions imply small covering number for the composition of functions.

Finally, we present two classical results about Rademacher complexity. For a class of real-valued functions $\mathcal{L}$ and dataset $\mathcal{X}_n$ with data size $n$, define the empirical Rademacher complexity of the function class $\mathcal{L}$ by:

$$\mathrm{Rad}_n(\mathcal{L}) = \mathop{\mathbb{E}}_{\alpha_i}\left[\sup_{l \in \mathcal{L}} \left(\frac{1}{n} \sum_{i=1}^n \alpha_i l(x_i)\right)\right],$$

where $x_i \in \mathcal{X}_n$, and $\alpha_i$'s are independent uniform $\pm 1$ random variables. Now we present the Rademacher complexity regression bounds.

**Lemma 6** (Rademacher complexity regression bounds, Mohri et al. (2018) Theorem 11.3 )**.** *Let $\ell : \mathcal{U} \subset \mathbb{R}^d \to \mathbb{R}$ be a nonnegative loss upper bounded by $M > 0$, and $\ell$ is $\mu$-Lipschitz for some $\mu > 0$, then for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of an i.i.d. sample $\mathcal{X}_n = \{x_1, x_2, \cdots, x_n\}$ of size $n$ according the distribution* $\Pr$*, the following holds for all $\ell \in \mathcal{L}$:*

$$\mathop{\mathbb{E}}_{x \sim P}[\ell(x)] \leq \frac{1}{n} \sum_{i=1}^n \ell(x_i) + 2\mu \,\mathrm{Rad}_n(\mathcal{L}) + 3M \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

**Lemma 7** (Dudley's entropy theorem, Dudley (1967))**.** *For any function class $\mathcal{L}$ containing functions $\ell : \mathcal{X} \to \mathbb{R}$, where $\forall x \in \mathcal{X}$ is bounded, we have that:*

$$\mathrm{Rad}_n(\mathcal{L}) \leq \inf_{\alpha > 0} \left(4\alpha + 12 \int_{\alpha}^{\sup_{\ell \in \mathcal{L}} \sqrt{\mathbb{E}_{x \sim P}[\ell^2]}} \sqrt{\frac{\ln \mathcal{N}(\mathcal{L}, \epsilon, L_2(P_n, |||\cdot|||))}{n}}\right).$$

**Corollary 1.** *Let us use $\widetilde{\mathcal{O}}(\cdot)$ to hide any factor that is sub-polynomial. Then we basically have that if* $\ln \mathcal{N}(\mathcal{L}, \epsilon, L_2(P_n, \|\!|\cdot|\!\|)) = \mathcal{O}^*\big(\frac{1}{\epsilon^p}\big)$*, then:*

$$\mathrm{Rad}_n(\mathcal{L}) = \begin{cases} \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) & p < 2, \\ \widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{n}}\right) & p = 2, \\ \widetilde{\mathcal{O}}\big(\frac{1}{n^{1/p}}\big) & p > 2. \end{cases}$$