Improving Covariance Conditioning of the SVD Meta-layer by Orthogonality

Yue Song[0000−0003−1573−5643], Nicu Sebe, and Wei Wang

DISI, University of Trento, Trento 38123, Italy
yue.song@unitn.it
https://github.com/KingJamesSong/OrthoImproveCond

Abstract. Inserting an SVD meta-layer into neural networks is prone to make the covariance ill-conditioned, which could harm the model in the training stability and generalization abilities. In this paper, we systematically study how to improve the covariance conditioning by enforcing orthogonality to the Pre-SVD layer. Existing orthogonal treatments on the weights are first investigated. However, these techniques can improve the conditioning but would hurt the performance. To avoid such a side effect, we propose the Nearest Orthogonal Gradient (NOG) and Optimal Learning Rate (OLR). The effectiveness of our methods is validated in two applications: decorrelated Batch Normalization (BN) and Global Covariance Pooling (GCP). Extensive experiments on visual recognition demonstrate that our methods can simultaneously improve the covariance conditioning and generalization. Moreover, the combinations with orthogonal weight can further boost the performances.

Keywords: Differentiable SVD, Covariance Conditioning, Orthogonality Constraint

1 Introduction

The Singular Value Decomposition (SVD) can factorize a matrix into orthogonal eigenbases and non-negative singular values, serving as an essential step for many matrix operations. Recently in computer vision and deep learning, many approaches integrated the SVD as a meta-layer in the neural networks to perform some differentiable spectral transformations, such as the matrix square root and inverse square root. The applications arise in a wide range of methods, including Global Covariance Pooling (GCP) [29, 44, 13], decorrelated Batch Normalization (BN) [20, 22, 45], Whitening an Coloring Transform (WCT) for universal style transfer [30, 8, 55], and Perspective-n-Point (PnP) problems [4, 6, 11].

For the input feature map $\mathbf{X}$ passed to the SVD meta-layer, one often first computes the covariance of the feature as $\mathbf{X}\mathbf{X}^T$. This can ensure that the covariance matrix is both symmetric and positive semi-definite, which does not involve any negative eigenvalues and leads to the identical left and right eigenvector matrices. However, it is observed that inserting the SVD layer into deep models would typically make the covariance very ill-conditioned [44], resulting in deleterious consequences on the stability and optimization of the training process. For a given covariance $\mathbf{A}$, its conditioning is measured by the condition number:

$$\kappa(\mathbf{A}) = \sigma_{max}(\mathbf{A})\sigma_{min}^{-1}(\mathbf{A}) \tag{1}$$

where $\sigma(\cdot)$ denotes the eigenvalue of the matrix. Mathematically speaking, the condition number measures how sensitive the SVD is to the errors of the input. Matrices with low condition numbers are considered **well-conditioned**, while matrices with high condition numbers are said to be **ill-conditioned**. Specific to neural networks, the ill-conditioned covariance matrices are harmful to the training process in several aspects, which we will analyze in detail later.

This phenomenon was first observed in the GCP methods by [44], and we found that it generally extrapolates to other SVD-related tasks, such as decorrelated BN. Fig. 1 depicts the covariance conditioning of these two tasks throughout the training. As can be seen, the integration of the SVD layer makes the generated covariance very ill-conditioned ($\approx 1e12$ for decorrelated BN and $\approx 1e16$ for GCP). By contrast, the conditioning of the approximate solver (Newton-Schulz iteration [19]) is about $1e5$ for decorrelated BN and is around $1e15$ for GCP, while the standard BN only has a condition number of $1e3$.

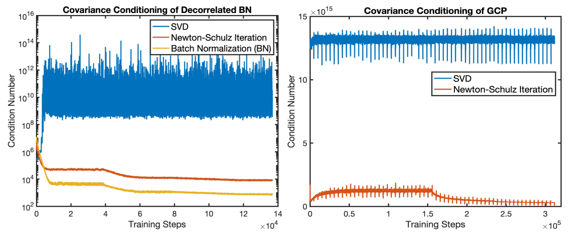

Fig. 1. The covariance conditioning of the SVD meta-layer during the training process in the tasks of decorrelated BN (*left*) and GCP (*Right*). The decorrelated BN is based on ResNet-50 and CIFAR100, while ImageNet and ResNet-18 are used for the GCP.

Ill-conditioned covariance matrices can harm the training of the network in both the forward pass (FP) and the backward pass (BP). For the FP, mainly the SVD solver is influenced in terms of stability and accuracy. Since the ill-conditioned covariance has many trivially-small eigenvalues, it is difficult for an SVD solver to accurately estimate them and large round-off errors are likely to be triggered, which might hurt the network performances. Moreover, the very imbalanced eigenvalue distribution can easily make the SVD solver fail to converge and cause the training failure [54, 44]. For the BP, as pointed out in [27, 56, 20], the feature covariance is closely related to the Hessian matrix during the backpropagation. Since the error curvature is given by the eigenvalues of the Hessian matrix [48], for the ill-conditioned Hessian, the Gradient Descent (GD) step would bounce back and forth in high curvature directions (large eigenvalues) and make slow progress in low curvature directions (small eigenvalues). As a consequence, the ill-conditioned covariance could cause slow convergence and oscillations in the optimization landscape. The generalization abilities of a deep model are thus harmed.

Due to the data-driven learning nature and the highly non-linear transform of deep neural networks, directly giving the analytical form of the covariance conditioning is intractable. Some simplifications have to be performed to ease the investigation. Since the covariance is generated and passed from the previous layer, the previous layer is likely to be the most relevant to the conditioning. Therefore, we naturally limit our focus to the Pre-SVD layer, *i.e.,* the layer before the SVD layer. To further simplify the analysis, we study the Pre-SVD layer in two consecutive training steps, which can be considered as a mimic of the whole training process. Throughout the paper, we mainly investigate some meaningful manipulations on the weight, the gradient, and the learning rate of the Pre-SVD layer in two sequential training steps. *Under our Pre-SVD layer simplifications, one promising direction to improve the conditioning is enforcing orthogonality on the weights.* Orthogonal weights have the norm-preserving property, which could improve the conditioning of the feature matrix. This technique has been widely studied in the literature of stable training and Lipschitz networks [33, 52, 43]. We select some representative methods and validate their effectiveness in the task of decorrelated BN. Our experiment reveals that these orthogonal techniques can greatly improve the covariance conditioning, but could only bring marginal performance improvements and even slight degradation. *This indicates that when the representation power of weight is limited, the improved conditioning does not necessarily lead to better performance. Orthogonalizing only the weight is thus insufficient to improve the generalization.*

Instead of seeking orthogonality constraints on the weights, we propose our Nearest Orthogonal Gradient (NOG) and Optimal Learning Rate (OLR). These two techniques explore the orthogonality possibilities about the learning rate and the gradient. More specifically, our NOG modifies the gradient of the Pre-SVD layer into its nearest-orthogonal form and keeps the GD direction unchanged. On the other hand, the proposed OLR dynamically changes the learning rate of the Pre-SVD layer at each training step such that the updated weight is as close to an orthogonal matrix as possible. The experimental results demonstrate that the proposed two techniques not only significantly improve the covariance conditioning but also bring obvious improvements in the validation accuracy of both GCP and decorrelated BN. Moreover, when combined with the orthogonal weight treatments, the performance can have further improvements.

The main contributions and findings are summarized below:

- We systematically study the problem of how to improve the covariance conditioning of the SVD meta-layer. We propose our Pre-SVD layer simplification to investigate this problem from the perspective of orthogonal constraints.
- We explore different techniques of orthogonal weights to improve the covariance conditioning. Our experiments reveal that these techniques could improve the conditioning but would harm the generalization abilities due to the limitation on the representation power of weight.

– We propose the nearest orthogonal gradient and optimal learning rate. The experiments on GCP and decorrelated BN demonstrate that these methods can attain better covariance conditioning and improved generalization. Their combinations with weight treatments can further boost the performance.

2 Related Work

In this section, we introduce the related work in differentiable matrix decomposition and the orthogonality in neural networks which could be relevant in improving the covariance conditioning.

2.1 Differentiable Matrix Decomposition

The differentiable matrix decomposition is widely used in neural networks as a spectral meta-layer. Ionescu *et al.* [24, 25] first propose the theory of matrix back-propagation and laid a foundation for the follow-up research. In deep neural networks, the transformation of matrix square root and its inverse are often desired due to the appealing spectral property. Their applications cover a wide range of computer vision tasks [45, 46]. To avoid the huge time consumption of the SVD, some iterative methods are also developed to approximate the solution [19, 45, 46]. In [20, 8, 23, 21, 22, 45], the inverse square root is used in the ZCA whitening transform to whiten the feature map, which is also known as the decorrelated BN. The Global Covariance Pooling (GCP) models [29, 28, 53, 58, 44, 13, 47] compute the matrix square root of the covariance as a spectral normalization, which achieves impressive performances on some recognition tasks, including large-scale visual classification [29, 44, 58, 45], fine-grained visual categorization [29, 28, 47], and video action recognition [13]. The Whitening and Coloring Transform (WCT), which uses both the matrix square root and inverse square root, is usually adopted in some image generation tasks such as neural style transfer [30, 55], image translation [51, 9], and domain adaptation [1, 10]. In the geometric vision problems, the differentiable SVD is usually applied to estimate the fundamental matrix and the camera pose [38, 11, 6]. Besides the SVD-based factorization, differentiating Cholesky decomposition [35] and some low-rank decomposition is used to approximate the attention mechanism [14, 59, 31] or to learn the constrained representations [7, 60].

2.2 Orthogonality in Neural Network

Orthogonal weights have the benefit of the norm-preserving property, *i.e.,* the relation $||\mathbf{WA}||_{\mathrm{F}}=||\mathbf{A}||_{\mathrm{F}}$ holds for any orthogonal $\mathbf{W}$. When it comes to deep neural networks, such a property can ensure that the signal stably propagates through deep networks without either exploding or vanishing gradients [3, 15], which could speed up convergence and encourage robustness and generalization. In general, there are three ways to enforce orthogonality to a layer: orthogonal weight initialization [40, 33, 57], orthogonal regularization [39, 2, 37, 2, 52], and explicit orthogonal weight via Carley transform or matrix exponential [32, 49, 43]. Among these techniques, orthogonal regularization and orthogonal weight are most commonly used as they often bring some practical improvements in generalization. Since the covariance is closely related to the weight matrix of the Pre-SVD layer, enforcing the orthogonality constraint could help to improve the covariance conditioning of the SVD meta-layer. We will choose some representative methods and validate their impact in Sec. 4.2.

Notice that the focus of existing literature is different from our work. The orthogonality constraints are often used to improve the Lipschitz constants of the neural network layers, which is expected to improve the visual quality in image generation [5, 34], to allow for better adversarial robustness [50, 43], and to improve generalization abilities [41, 52]. Our work is concerned with improving the covariance conditioning and generalization performance. Moreover, the orthogonality literature mainly investigates how to enforce orthogonality to weight matrices, whereas less attention is put on the gradient and learning rate. In Sec. 5, we will explore such possibilities and propose our solutions: nearest orthogonal gradient and optimal learning rate which is optimal in the sense that the updated weight is as close to an orthogonal matrix as possible.

3 Background: SVD Meta-Layer

This section presents the background knowledge about the propagation rules of the SVD meta-layer.

3.1 Forward Pass

Given the reshape feature $\mathbf{X}{\in}\mathbb{R}^{d\times N}$ where d denotes the feature dimensionality (*i.e.,* the number of channels) and N represents the number of features (*i.e.,* the product of spatial dimensions of features), an SVD meta-layer first computes the sample covariance as:

$$\mathbf{P} = \mathbf{X}\mathbf{J}\mathbf{X}^{T}, \mathbf{J} = \frac{1}{N}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^{T}) \tag{2}$$

where $\mathbf{J}$ represents the centering matrix, $\mathbf{I}$ denotes the identity matrix, and $\mathbf{1}$ is a column vector whose values are all ones, respectively. The covariance is always positive semi-definite (PSD) and does not have any negative eigenvalues. Afterward, the eigendecomposition is performed using the SVD:

$$\mathbf{P} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{T},\ \mathbf{\Lambda} = \mathrm{diag}(\lambda_{1}, \dots, \lambda_{\mathrm{d}}) \tag{3}$$

where $\mathbf{U}$ is the orthogonal eigenvector matrix, $\mathrm{diag}(\cdot)$ denotes transforming a vector to a diagonal matrix, and $\mathbf{\Lambda}$ is the diagonal matrix in which the eigenvalues are sorted in a non-increasing order *i.e.,* $\lambda_i{\geq}\lambda_{i+1}$. Then depending on the application, the matrix square root or the inverse square root is calculated as:

$$\begin{aligned} \mathbf{Q} \triangleq \mathbf{P}^{\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^{T}, \mathbf{\Lambda}^{\frac{1}{2}} = \mathrm{diag}(\lambda_{1}^{\frac{1}{2}}, \dots, \lambda_{d}^{\frac{1}{2}}) \\ \mathbf{S} \triangleq \mathbf{P}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^{T}, \mathbf{\Lambda}^{-\frac{1}{2}} = \mathrm{diag}(\lambda_{1}^{-\frac{1}{2}}, \dots, \lambda_{d}^{-\frac{1}{2}}) \end{aligned} \tag{4}$$

The matrix square root $\mathbf{Q}$ is often used in GCP-related tasks [29, 58, 44], while the application of decorrelated BN [20, 42] widely applies the inverse square root $\mathbf{S}$. In certain applications such as WCT, both $\mathbf{Q}$ and $\mathbf{S}$ are required.

3.2 Backward Pass

Let $\frac{\partial l}{\partial \mathbf{Q}}$ and $\frac{\partial l}{\partial \mathbf{S}}$ denote the partial derivative of the loss l w.r.t to the matrix square root $\mathbf{Q}$ and the inverse square root $\mathbf{S}$, respectively. Then the gradient passed to the eigenvector is computed as:

$$\frac{\partial l}{\partial \mathbf{U}}\Big|_{\mathbf{Q}} = (\frac{\partial l}{\partial \mathbf{Q}} + (\frac{\partial l}{\partial \mathbf{Q}})^T)\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}},\ \frac{\partial l}{\partial \mathbf{U}}\Big|_{\mathbf{S}} = (\frac{\partial l}{\partial \mathbf{S}} + (\frac{\partial l}{\partial \mathbf{S}})^T)\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{5}$$

Notice that the gradient equations for $\mathbf{Q}$ and $\mathbf{S}$ are different. For the eigenvalue, the gradient is calculated as:

$$\frac{\partial l}{\partial \mathbf{\Lambda}}\Big|_{\mathbf{Q}} = \frac{1}{2}\mathrm{diag}(\lambda_1^{-\frac{1}{2}}, \dots, \lambda_{\mathrm{d}}^{-\frac{1}{2}})\mathbf{U}^{\mathrm{T}}\frac{\partial l}{\partial \mathbf{Q}}\mathbf{U},\ \frac{\partial \mathrm{l}}{\partial \mathbf{\Lambda}}\Big|_{\mathbf{S}} = -\frac{1}{2}\mathrm{diag}(\lambda_1^{-\frac{3}{2}}, \dots, \lambda_{\mathrm{d}}^{-\frac{3}{2}})\mathbf{U}^{\mathrm{T}}\frac{\partial l}{\partial \mathbf{S}}\mathbf{U} \tag{6}$$

Subsequently, the derivative of the SVD step can be calculated as:

$$\frac{\partial l}{\partial \mathbf{P}} = \mathbf{U}((\mathbf{K}^T \circ (\mathbf{U}^T\frac{\partial l}{\partial \mathbf{U}})) + (\frac{\partial l}{\partial \mathbf{\Lambda}})_{\mathrm{diag}})\mathbf{U}^T \tag{7}$$

where $\circ$ denotes the matrix Hadamard product, and the matrix $\mathbf{K}$ consists of entries $K_{ij}{=}1/(\lambda_i{-}\lambda_j)$ if $i{\neq}j$ and $K_{ij}{=}0$ otherwise. This step is the same for both $\mathbf{Q}$ and $\mathbf{S}$. Finally, we have the gradient passed to the feature $\mathbf{X}$ as:

$$\frac{\partial l}{\partial \mathbf{X}} = (\frac{\partial l}{\partial \mathbf{P}} + (\frac{\partial l}{\partial \mathbf{P}})^T)\mathbf{X}\mathbf{J} \tag{8}$$

With the above rules, the SVD function can be easily inserted into any neural networks and trained end-to-end as a meta-layer.

4 Pre-SVD Layer and Weight Treatments

In this section, we first motivate our simplification of the Pre-SVD layer, and then validate the efficacy of some representative weight treatments.

4.1 Pre-SVD Layer Simplification

The neural network consists of a sequential of non-linear layers where the learning of each layer is data-driven. Stacking these layers leads to a highly non-linear and complex transform, which makes directly analyzing the covariance conditioning intractable. To solve this issue, we have to perform some simplifications.

Our simplifications involve limiting the analysis only to the layer previous to the SVD layer (which we dub as the Pre-SVD layer) in two consecutive training steps. The Pre-SVD layer directly determines the conditioning of the generated covariance, while the two successive training steps are a mimic of the whole training process. The idea is to simplify the complex transform by analyzing the sub-model (two layers) and the sub-training (two steps), which can be considered as an "abstract representation" of the deep model and its complete training.

Let $\mathbf{W}$ denote the weight matrix of the Pre-SVD layer. Then for the input $\mathbf{X}_l$ passed to the layer, we have:

$$\mathbf{X}_{l+1} = \mathbf{W}\mathbf{X}_l + \mathbf{b} \tag{9}$$

where $\mathbf{X}_{l+1}$ is the feature passed to the SVD layer, and $\mathbf{b}$ is the bias vector. Since the bias $\mathbf{b}$ has a little influence here, we can sufficiently omit it for simplicity. The covariance in this step is computed as $\mathbf{W}\mathbf{X}_l\mathbf{X}_l^T\mathbf{W}^T$. After the BP, the weight matrix is updated as $\mathbf{W}-\eta\frac{\partial l}{\partial \mathbf{W}}$ where η denotes the learning rate of the layer. Let $\mathbf{Y}_l$ denote the passed-in feature of the next training step. Then the covariance is calculated as:

$$\begin{aligned}
\mathbf{C} &= \Big((\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}}) \cdot \mathbf{Y}_l\Big)\Big((\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}}) \cdot \mathbf{Y}_l\Big)^T \\
&= (\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}})\mathbf{Y}_l\mathbf{Y}_l^T(\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}})^T \\
&= \mathbf{W}\mathbf{Y}_l\mathbf{Y}_l^T\mathbf{W}^T - \eta\frac{\partial l}{\partial \mathbf{W}}\mathbf{Y}_l\mathbf{Y}_l^T\mathbf{W}^T - \eta\mathbf{W}\mathbf{Y}_l\mathbf{Y}_l^T(\frac{\partial l}{\partial \mathbf{W}})^T + \eta^2\frac{\partial l}{\partial \mathbf{W}}\mathbf{Y}_l\mathbf{Y}_l^T(\frac{\partial l}{\partial \mathbf{W}})^T
\end{aligned} \tag{10}$$

where $\mathbf{C}$ denotes the generated covariance of the second step. Now the problem becomes how to stop the new covariance $\mathbf{C}$ from becoming worse-conditioned than $\mathbf{W}\mathbf{X}_l\mathbf{X}_l^T\mathbf{W}^T$. In eq. (10), three variables could influence the conditioning: the weight $\mathbf{W}$, the gradient of the last step $\frac{\partial l}{\partial \mathbf{W}}$, and the learning rate η of this layer. Among them, the weight $\mathbf{W}$ seems to be the most important as it contributes to three terms of eq. (10). Moreover, the first term $\mathbf{W}\mathbf{Y}_l\mathbf{Y}_l^T\mathbf{W}^T$ computed by $\mathbf{W}$ is not attenuated by η or η^2 like the other terms. Therefore, it is natural to first consider manipulating $\mathbf{W}$ such that the conditioning of $\mathbf{C}$ could be improved.

4.2 General Treatments on Weights

In the literature of enforcing orthogonality to the neural network, there are several techniques to improve the conditioning of the weight $\mathbf{W}$. Now we introduce some representatives methods and validate their impacts.

Spectral Normalization (SN). In [34], the authors propose a normalization method to stabilize the training of generative models [16] by dividing the weight matrix with its largest eigenvalue. The process is defined as:

$$\mathbf{W}/\sigma_{max}(\mathbf{W}) \tag{11}$$

Such a normalization can ensure that the spectral radius of $\mathbf{W}$ is always 1, *i.e.,* $\sigma_{max}(\mathbf{W})$=1. This could help to reduce the conditioning of the covariance since we have $\sigma_{max}(\mathbf{W}\mathbf{Y}_l)$=$\sigma_{max}(\mathbf{Y}_l)$ after the spectral normalization.

Orthogonal Loss (OL). Besides limiting the spectral radius of $\mathbf{W}$, enforcing orthogonality constraint could also improve the covariance conditioning. As orthogonal matrices are norm-preserving (*i.e.,* $||\mathbf{W}\mathbf{Y}_l||_{\mathrm{F}}{=}||\mathbf{W}||_{\mathrm{F}}$), lots of methods have been proposed to encourage orthogonality on weight matrices for more stable training and better signal-preserving property [36, 2, 52, 49, 43]. One common technique is to apply *soft* orthogonality [52] by the following regularization:

$$l = ||\mathbf{W}\mathbf{W}^T - \mathbf{I}||_{\mathrm{F}} \tag{12}$$

This extra loss is added in the optimization objective to encourage more orthogonal weight matrices. However, since the constraint is achieved by regularization, the weight matrix is not exactly orthogonal at each training step.

Orthogonal Weights (OW). Instead of applying *soft* orthogonality by regularization, some methods can explicitly enforce *hard* orthogonality to the weight matrices [49, 43]. The technique of [43] is built on the mathematical property: for any skew-symmetric matrix, its matrix exponential is an orthogonal matrix.

$$\exp(\mathbf{W} - \mathbf{W}^T)\exp(\mathbf{W} - \mathbf{W}^T)^T = \mathbf{I} \tag{13}$$

where the operation of $\mathbf{W}{-}\mathbf{W}^T$ is to make the matrix skew-symmetric, *i.e.,* the relation $\mathbf{W}{-}\mathbf{W}^T{=}-(\mathbf{W}{-}\mathbf{W}^T)^T$ always holds. Then $\exp(\mathbf{W}{-}\mathbf{W}^T)$ is used as the weight. This technique explicitly constructs the weight as an orthogonal matrix. The orthogonal constraint is thus always satisfied during the training.

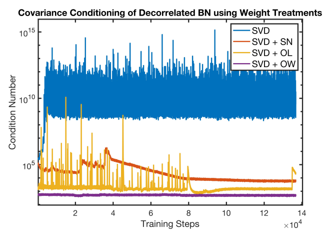

Fig. 2. The covariance conditioning during the training process. All the weight treatments can improve the conditioning.

Table 1. Performance of different weight treatments on ResNet-50 and CIFAR100 based on 10 runs.

Methods	mean±std	min
SVD	19.99±0.16	19.80
SVD + SN	19.94±0.33	19.60
SVD + OL	**19.73±0.28**	**19.54**
SVD + OW	20.06±0.17	19.94
Newton-Schulz iteration	19.45±0.33	19.01

We apply the above three techniques in the experiment of decorrelated BN. Fig. 2 displays the covariance conditioning throughout the training, and Table 1 presents the corresponding validation errors. As can be seen, all of these techniques attain much better conditioning, but the performance improvements are not encouraging. The SN reduces the conditioning to around 10^5, while the validation error marginally improves. The *soft* orthogonality by the OL brings slight improvement on the performance despite some variations in the conditioning. The conditioning variations occur because the orthogonality constraint by regularization is not strictly enforced. Among the weight treatments, the *hard* orthogonality by the OW achieves the best covariance conditioning, continuously maintaining the condition number around 10^3 throughout the training. However, the OW slightly hurts the validation error. This implies that better covariance conditioning does not necessarily correspond to the improved performance, and orthogonalizing only the weight cannot improve the generalization. *We conjecture that enforcing strict orthogonality only on the weight might limit its representation power.* Nonetheless, as will be discussed in Sec. 5.1, the side effect can be canceled when we simultaneously orthogonalize the gradient.

5 Nearest Orthogonal Gradient & Optimal Learning Rate

In this section, we introduce our proposed two techniques on modifying the gradient and learning rate of the Pre-SVD layer. Their combinations with the weight treatments are also discussed.

5.1 Nearest Orthogonal Gradient (NOG)

As discussed in Sec. 4.1, the covariance conditioning is also influenced by the gradient $\frac{\partial l}{\partial \mathbf{W}}$. However, existing literature mainly focuses on orthogonalizing the weights. To make the gradient also orthogonal, we propose to find the nearest-orthogonal gradient of the Pre-SVD layer. Different matrix nearness problems have been studied in [18], and the nearest-orthogonal problem is defined as:

$$\min_{\mathbf{R}} ||\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R}||_{\mathrm{F}} \ \textit{subject to}\ \mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{14}$$

where $\mathbf{R}$ is the seeking solution. To obtain such an orthogonal matrix, we can construct the error function as:

$$e(\mathbf{R}) = Tr\Big((\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R})^T(\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R})\Big) + Tr\Big(\mathbf{\Sigma}\mathbf{R}^T\mathbf{R} - \mathbf{I}\Big) \tag{15}$$

where $Tr(\cdot)$ is the trace measure, and $\mathbf{\Sigma}$ denotes the symmetric matrix Lagrange multiplier. The closed-form solution is given by:

$$\mathbf{R} = \frac{\partial l}{\partial \mathbf{W}}\Big((\frac{\partial l}{\partial \mathbf{W}})^T \frac{\partial l}{\partial \mathbf{W}}\Big)^{-\frac{1}{2}} \tag{16}$$

The detailed derivation is given in the supplementary material. If we have the SVD of the gradient ($\mathbf{U}\mathbf{S}\mathbf{V}^T{=}\frac{\partial l}{\partial \mathbf{W}}$), the solution can be further simplified as:

$$\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T(\mathbf{V}\mathbf{S}^{-1}\mathbf{V}^T) = \mathbf{U}\mathbf{V}^T \tag{17}$$

As indicated above, the nearest orthogonal gradient is achieved by setting the singular value matrix to the identity matrix, *i.e.,* setting $\mathbf{S}$ to $\mathbf{I}$. Notice that only the gradient of Pre-SVD layer is changed, while that of the other layers is not modified. Our proposed NOG can bring several practical benefits.

Orthogonal Constraint and Optimal Conditioning. The orthogonal constraint is exactly enforced on the gradient as we have $(\mathbf{U}\mathbf{V}^T)^T\mathbf{U}\mathbf{V}^T{=}\mathbf{I}$. Since we explicitly set all the singular values to 1, the optimal conditioning is also achieved, *i.e.,* $\kappa(\frac{\partial l}{\partial \mathbf{W}}){=}1$. This could help to improve the conditioning.

Keeping Gradient Descent Direction Unchanged. In the high-dimensional optimization landscape, the many curvature directions (GD directions) are characterized by the eigenvectors of gradient ($\mathbf{U}$ and $\mathbf{V}$). Although our modification changes the gradient, the eigenvectors and the GD directions are untouched. In other words, our NOG only adjusts the step size in each GD direction. This indicates that the modified gradients will not harm the network performances.

Combination with Weight Treatments. Our orthogonal gradient and the previous weight treatments are complementary. They can be jointly used to simultaneously orthogonalize the gradient and weight. In the following, we will validate their joint impact on the conditioning and performance.

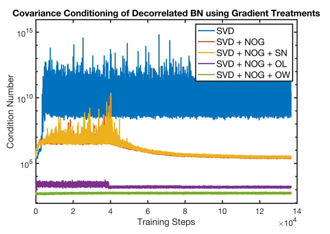

Fig. 3. The covariance conditioning during the training process using orthogonal gradient and combined weight treatments.

Table 2. Performance of gradient and weight treatments on ResNet-50 and CIFAR100. Each result is based on 10 runs.

Methods	mean±std	min
SVD	19.99±0.16	19.80
SVD + NOG	19.43±0.24	19.15
SVD + NOG + SN	19.43±0.21	19.20
SVD + NOG + OL	20.14±0.39	19.54
SVD + NOG + OW	**19.22±0.28**	**18.90**
Newton-Schulz iteration	19.45±0.33	19.01

Fig. 3 and Table 2 present the covariance conditioning of decorrelated BN and the corresponding validation errors, respectively. As we can observe, solely using the proposed NOG can largely improve the covariance conditioning, decreasing the condition number from 10^{12} to 10^{6}. Though this improvement is not as significant as the orthogonal constraints (*e.g.,* OL and OW), our NOG can benefit more the generalization abilities, leading to the improvement of validation error by 0.6%. Combining the SN with our NOG does not lead to obvious improvements in either the conditioning or validation errors, whereas the joint use of NOG and OL harms the network performances. This is because the orthogonality constraint by loss might not be enforced under the gradient manipulation. When our NOG is combined with the OW, the side effect of using only OW is eliminated and the performance is further boosted by 0.3%. This phenomenon demonstrates that when the gradient is orthogonal, applying the orthogonality constraint to the weight could also be beneficial to the generalization.

5.2 Optimal Learning Rate (OLR)

So far, we only consider orthogonalizing $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ separately, but how to jointly optimize $\mathbf{W}-\eta\frac{\partial l}{\partial \mathbf{W}}$ has not been studied yet. Actually, it is desired to choose an appropriate learning rate η such that the updated weight is close to an orthogonal matrix. To this end, we need to achieve the following objective:

$$\min_{\eta} ||(\mathbf{W}-\eta\frac{\partial l}{\partial \mathbf{W}})(\mathbf{W}-\eta\frac{\partial l}{\partial \mathbf{W}})^T-\mathbf{I}||_{\mathrm{F}} \tag{18}$$

This optimization problem can be more easily solved in the vector form. Let $\mathbf{w}$, $\mathbf{i}$, and $\mathbf{l}$ denote the vectorized $\mathbf{W}$, $\mathbf{I}$, and $\frac{\partial l}{\partial \mathbf{W}}$, respectively. Then we construct the error function as:

$$e(\eta) = \Big((\mathbf{w}-\eta\mathbf{l})^T(\mathbf{w}-\eta\mathbf{l})-\mathbf{i}\Big)^T\Big((\mathbf{w}-\eta\mathbf{l})^T(\mathbf{w}-\eta\mathbf{l})-\mathbf{i}\Big) \tag{19}$$

Expanding and differentiating the equation w.r.t. η lead to:

$$\begin{aligned}\frac{de(\eta)}{d\eta} &\approx -4\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{w}+4\eta\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{l}+8\eta\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}=0\\ \eta^{\star} &\approx \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l}+2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}\end{aligned} \tag{20}$$

where some higher-order terms are neglected. The detailed derivation is given in the supplementary material. Though the proposed OLR yields the updated weight nearest to an orthogonal matrix theoretically, the value of $\eta^{\star}$ is unbounded for arbitrary $\mathbf{w}$ and $\mathbf{l}$. Directly using $\eta^{\star}$ might cause unstable training. To avoid this issue, we propose to use the OLR only when its value is smaller than the learning rate of other layers. Let lr denote the learning rate of the other layers. The switch process can be defined as:

$$\eta = \begin{cases}\eta^{\star} & if\ \eta^{\star} < lr\\ lr & otherwise\end{cases} \tag{21}$$

Combination with Weight/Gradient Treatments. When either the weight or the gradient is orthogonal, our OLR needs to be carefully used. When only $\mathbf{W}$ is orthogonal, $\mathbf{w}^T\mathbf{w}$ is a small constant and it is very likely to have $\mathbf{w}^T\mathbf{w}{\ll}\mathbf{l}^T\mathbf{w}$. Consequently, we have $\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w}{\ll}\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}$ and $\eta^{\star}$ will attenuate to zero. Similarly for orthogonal gradient, we have $\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w}{\ll}\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{l}$ and this will cause $\eta^{\star}$ close to zero. Therefore, the proposed OLR cannot work when either the weight or gradient is orthogonal. Nonetheless, we note that if both $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ are orthogonal, our $\eta^{\star}$ is bounded. Specifically, we have:

Proposition 1 *When both* $\mathbf{W}$ *and* $\frac{\partial l}{\partial \mathbf{W}}$ *are orthogonal,* $\eta^{\star}$ *is both upper and lower bounded. The upper bound is* $\frac{N^2}{N^2+2}$ *and the lower bound is* $\frac{1}{N^2+2}$ *where* N *denotes the row dimension of* $\mathbf{W}$.

We give the detailed proof in the supplementary material. Obviously, the upper bound of $\eta^{\star}$ is smaller than 1. For the lower bound, since the row dimension of N is often large (*e.g.,* 64), the lower bound of $\eta^{\star}$ can be according very small (*e.g.,* $2e{-}4$). This indicates that our proposed OLR could also give a small learning rate even in the later stage of the training process.

In summary, the optimal learning rate is set such that the updated weight is optimal in the sense that it become as close to an orthogonal matrix as possible. In particular, it is suitable when both the gradient and weight are orthogonal.

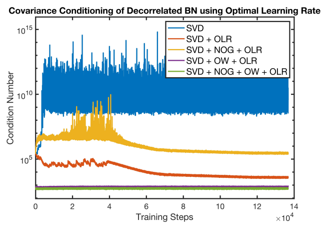

Fig. 4. The covariance conditioning during the training process using optimal learning rate and hybrid treatments.

Table 3. Performance of optimal learning rate and hybrid treatments on ResNet-50 and CIFAR100 based on 10 runs.

Methods	mean±std	min
SVD	19.99±0.16	19.80
SVD + OLR	19.50±0.39	18.95
SVD + NOG + OLR	19.77±0.27	19.36
SVD + OW + OLR	20.61±0.22	20.43
SVD + NOG + OW +OLR	**19.05±0.31**	**18.77**
Newton-Schulz iteration	19.45±0.33	19.01

We give the covariance conditioning and the validation errors of our OLR in Fig. 4 and in Table 3, respectively. Our proposed OLR significantly reduces the condition number to 10^4 and improves the validation error by 0.5%. When combined with either orthogonal weight or orthogonal gradient, there is a slight degradation on the validation errors. This meets our expectation as $\eta^{\star}$ would attenuate to zero in both cases. However, when both $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ are orthogonal, jointly using our OLR achieves the best performance, outperforming only OLR by 0.5% and beating OW+NOG by 0.2%. This observation confirms that the proposed OLR works well for simultaneously orthogonal $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$.

6 Experiments

We validate the proposed approaches in two applications: GCP and decorrelated BN. These two tasks are very representative because they have different usages of the SVD meta-layer. The GCP uses the matrix square root, while the decorrelated BN applies the inverse square root. In addition, the models of decorrelated BN often insert the SVD meta-layer at the beginning of the network, whereas the GCP models integrate the layer before the FC layer.

6.1 Decorrelated Batch Normalization

Table 4 compares the performance of each method on CIFAR10/CIFAR100 [26] based on ResNet-50 [17]. Both of our NOG and OLR achieve better performance

Table 4. Performance comparison of different decorrelated BN methods on CIFAR10/CIFAR100 [26] based on ResNet-50 [17]. We report each result based on 10 runs. The best four results are highlighted in **red**, **blue**, **green**, and **cyan** respectively.

Methods	CIFAR10		CIFAR100	
	mean±std	min	mean±std	min
SVD	4.35±0.09	4.17	19.99±0.16	19.80
SVD + Spectral Norm (SN)	4.31±0.10	4.15	19.94±0.33	19.60
SVD + Orthogonal Loss (OL)	4.28±0.07	4.23	19.73±0.28	19.54
SVD + Orthogonal Weight (OW)	4.42±0.09	4.28	20.06±0.17	19.94
SVD + Nearest Orthogonal Gradient (NOG)	**4.15±0.06**	**4.04**	**19.43±0.24**	**19.15**
SVD + Optimal Learning Rate (OLR)	**4.23±0.17**	**3.98**	**19.50±0.39**	**18.95**
SVD + NOG + OW	**4.09±0.07**	**4.01**	**19.22±0.28**	**18.90**
SVD + NOG + OW + OLR	**3.93±0.09**	**3.85**	**19.05±0.31**	**18.77**
Newton-Schulz iteration	4.20±0.11	4.11	19.45±0.33	19.01

than other weight treatments and the SVD. Moreover, when hybrid treatments are adopted, we can observe step-wise steady improvements on the validation errors. Among these techniques, the joint usage of OLR with NOG and OW achieves the best performances across metrics and datasets, outperforming the SVD baseline by 0.4% on CIFAR10 and by 0.9% on CIFAR100. This demonstrates that these treatments are complementary and can benefit each other.

Table 5. Performance comparison of different GCP methods on ImageNet [12] based on ResNet-18 [17]. The failure times denote the total times of non-convergence of the SVD solver during one training process. The best four results are highlighted in **red**, **blue**, **green**, and **cyan** respectively.

Method	Failure Times	Top-1 Acc. (%)	Top-5 Acc. (%)
SVD	5	73.13	91.02
SVD + Spectral Norm (SN)	2	73.28 (↑ 0.2)	91.11 (↑ 0.1)
SVD + Orthogonal Loss (OL)	1	71.75 (↓ 1.4)	90.20 (↓ 0.8)
SVD + Orthogonal Weight (OW)	2	73.07 (↓ 0.1)	90.93 (↓ 0.1)
SVD + Nearest Orthogonal Gradient (NOG)	1	**73.51** (↑ **0.4**)	**91.35** (↑ **0.3**)
SVD + Optimal Learning Rate (OLR)	0	**73.39** (↑ **0.3**)	**91.26** (↑ **0.2**)
SVD + NOG + OW	0	**73.71** (↑ **0.6**)	**91.43** (↑ **0.4**)
SVD + NOG + OW + OLR	0	**73.82** (↑ **0.7**)	**91.57** (↑ **0.6**)
Newton-Schulz iteration	0	73.36 (↑ 0.2)	90.96 (↓ 0.1)

6.2 Global Covariance Pooling

Table 5 presents the total failure times of the SVD solver in one training process and the validation accuracy on ImageNet [12] based on ResNet-18 [17]. The results are very coherent with our experiment of decorrelated BN. Among the weight treatments, the OL and OW hurt the performance, while the SN improves that of SVD by 0.2%. Our proposed NOG and OLR outperform the weight treatments and improve the SVD baseline by 0.4% and by 0.3%, respectively. Moreover, the combinations with the orthogonal weight further boost the performance. Specifically, combining NOG and OW surpasses the SVD by 0.6%. The joint use of OW with NOG and OLR achieves the best performance among all the methods and beats the SVD by 0.7%.

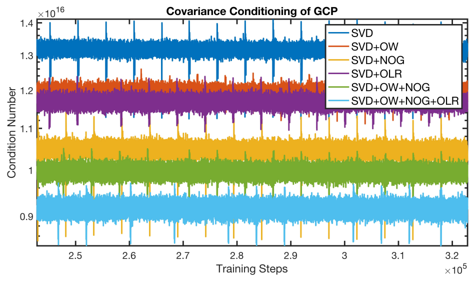

Fig. 5. The covariance conditioning of GCP methods in the later stage of the training. The periodic spikes are caused by the evaluation on the validation set after every epoch.

Fig. 5 depicts the covariance conditioning in the later training stage. Our OLR and the OW both reduce the condition number by around $1e15$, whereas the proposed NOG improves the condition number by $2e15$. When hybrid treatments are used, combining NOG and OW attains better conditioning than the separate usages. Furthermore, simultaneously using all the techniques leads to the best conditioning and improves the condition number by $5e15$.

The covariance conditioning of GCP tasks is not improved as much as that of decorrelated BN. This might stem from the unique architecture of GCP models: the covariance is directly used as the final representation and fed to the FC layer. We conjecture that this setup might cause the covariance to have a high condition number. The approximate solver (Newton-Schulz iteration) does not have well-conditioned matrices either ($\approx 1e15$), which partly supports our conjecture.

7 Conclusion and Future Work

In this paper, we explore different approaches to improve the covariance conditioning of the SVD meta-layer. Existing treatments on orthogonal weight are first studied. Our experiments reveal that these techniques could improve the conditioning but might hurt the performance due to the limitation on the representation power. To avoid the side effect of orthogonal weight, we propose the nearest orthogonal gradient and the optimal learning rate, both of which could simultaneously attain better covariance conditioning and improved generalization abilities. Moreover, their combinations with orthogonal weight further boost the performance. In future work, we would like to study the problem of ill-conditioned covariance from other perspectives and extend our proposed techniques to other SVD-related methods.

Acknowledgement This research was supported by the EU H2020 projects AI4Media (No. 951911) and SPRING (No. 871245).

References

1. Abramov, A., Bayer, C., Heller, C.: Keep it simple: Image statistics matching for domain adaptation. arXiv preprint arXiv:2005.12551 (2020)
2. Bansal, N., Chen, X., Wang, Z.: Can we gain more from orthogonality regularizations in training deep networks? In: NeurIPS (2018)
3. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks (1994)
4. Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C.: Dsac-differentiable ransac for camera localization. In: CVPR (2017)
5. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. In: ICLR (2019)
6. Campbell, D., Liu, L., Gould, S.: Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In: ECCV (2020)
7. Chan, T.H., Jia, K., Gao, S., Lu, J., Zeng, Z., Ma, Y.: Pcanet: A simple deep learning baseline for image classification? IEEE TIP (2015)
8. Chiu, T.Y.: Understanding generalized whitening and coloring transform for universal style transfer. In: ICCV (2019)
9. Cho, W., Choi, S., Park, D.K., Shin, I., Choo, J.: Image-to-image translation via group-wise deep whitening-and-coloring transformation. In: CVPR (2019)
10. Choi, S., Jung, S., Yun, H., Kim, J.T., Kim, S., Choo, J.: Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In: CVPR (2021)
11. Dang, Z., Yi, K.M., Hu, Y., Wang, F., Fua, P., Salzmann, M.: Eigendecomposition-free training of deep networks for linear least-square problems. TPAMI (2020)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
13. Gao, Z., Wang, Q., Zhang, B., Hu, Q., Li, P.: Temporal-attentive covariance pooling networks for video recognition. In: NeurIPS (2021)
14. Geng, Z., Guo, M.H., Chen, H., Li, X., Wei, K., Lin, Z.: Is attention better than matrix decomposition? In: ICLR (2021)
15. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS (2010)
16. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. NeurIPS (2014)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
18. Higham, N.J.: Matrix nearness problems and applications. Citeseer (1988)
19. Higham, N.J.: Functions of matrices: theory and computation. SIAM (2008)
20. Huang, L., Yang, D., Lang, B., Deng, J.: Decorrelated batch normalization. In: CVPR (2018)
21. Huang, L., Zhao, L., Zhou, Y., Zhu, F., Liu, L., Shao, L.: An investigation into the stochasticity of batch whitening. In: CVPR (2020)
22. Huang, L., Zhou, Y., Liu, L., Zhu, F., Shao, L.: Group whitening: Balancing learning efficiency and representational capacity. In: CVPR (2021)
23. Huang, L., Zhou, Y., Zhu, F., Liu, L., Shao, L.: Iterative normalization: Beyond standardization towards efficient whitening. In: CVPR (2019)
24. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. In: ICCV (2015)

25. Ionescu, C., Vantzos, O., Sminchisescu, C.: Training deep networks with structured layers by matrix backpropagation. arXiv preprint arXiv:1509.07838 (2015)
26. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Tront (2009)
27. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade, pp. 9–48. Springer (2012)
28. Li, P., Xie, J., Wang, Q., Gao, Z.: Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In: CVPR (2018)
29. Li, P., Xie, J., Wang, Q., Zuo, W.: Is second-order information helpful for large-scale visual recognition? In: ICCV (2017)
30. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: NeurIPS (2017)
31. Lu, J., Yao, J., Zhang, J., Zhu, X., Xu, H., Gao, W., Xu, C., Xiang, T., Zhang, L.: Soft: softmax-free transformer with linear complexity. NeurIPS (2021)
32. Maduranga, K.D., Helfrich, K.E., Ye, Q.: Complex unitary recurrent neural networks using scaled cayley transform. In: AAAI (2019)
33. Mishkin, D., Matas, J.: All you need is a good init. ICLR (2016)
34. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: ICLR (2018)
35. Murray, I.: Differentiation of the cholesky decomposition. arXiv preprint arXiv:1602.07527 (2016)
36. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: ICML (2013)
37. Qi, H., You, C., Wang, X., Ma, Y., Malik, J.: Deep isometric learning for visual recognition. In: ICML. PMLR (2020)
38. Ranftl, R., Koltun, V.: Deep fundamental matrix estimation. In: ECCV (2018)
39. Rodríguez, P., Gonzalez, J., Cucurull, G., Gonfaus, J.M., Roca, X.: Regularizing cnns with locally constrained decorrelations. In: ICLR (2016)
40. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In: ICLR (2014)
41. Sedghi, H., Gupta, V., Long, P.M.: The singular values of convolutional layers. In: ICLR (2018)
42. Siarohin, A., Sangineto, E., Sebe, N.: Whitening and coloring batch transform for gans. In: ICLR (2018)
43. Singla, S., Feizi, S.: Skew orthogonal convolutions. In: ICML (2021)
44. Song, Y., Sebe, N., Wang, W.: Why approximate matrix square root outperforms accurate svd in global covariance pooling? In: ICCV (2021)
45. Song, Y., Sebe, N., Wang, W.: Fast differentiable matrix square root. In: ICLR (2022)
46. Song, Y., Sebe, N., Wang, W.: Fast differentiable matrix square root and inverse square root. arXiv preprint arXiv:2201.12543 (2022)
47. Song, Y., Sebe, N., Wang, W.: On the eigenvalues of global covariance pooling for fine-grained visual recognition. IEEE TPAMI (2022)
48. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: ICML (2013)
49. Trockman, A., Kolter, J.Z.: Orthogonalizing convolutional layers with the cayley transform. In: ICLR (2020)
50. Tsuzuku, Y., Sato, I., Sugiyama, M.: Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. NeurIPS (2018)

51. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: CVPR (2017)
52. Wang, J., Chen, Y., Chakraborty, R., Yu, S.X.: Orthogonal convolutional neural networks. In: CVPR (2020)
53. Wang, Q., Xie, J., Zuo, W., Zhang, L., Li, P.: Deep cnns meet global covariance pooling: Better representation and generalization. TPAMI (2020)
54. Wang, W., Dang, Z., Hu, Y., Fua, P., Salzmann, M.: Robust differentiable svd. TPAMI (2021)
55. Wang, Z., Zhao, L., Chen, H., Qiu, L., Mo, Q., Lin, S., Xing, W., Lu, D.: Diversified arbitrary style transfer via deep feature perturbation. In: CVPR (2020)
56. Wiesler, S., Ney, H.: A convergence analysis of log-linear training. NeurIPS (2011)
57. Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., Pennington, J.: Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In: ICML. PMLR (2018)
58. Xie, J., Zeng, R., Wang, Q., Zhou, Z., Li, P.: So-vit: Mind visual tokens for vision transformer. arXiv preprint arXiv:2104.10935 (2021)
59. Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., Singh, V.: Nyströmformer: A nyström-based algorithm for approximating self-attention. In: AAAI (2021)
60. Yang, Y., Sun, J., Li, H., Xu, Z.: Admm-net: A deep learning approach for compressive sensing mri. arXiv preprint arXiv:1705.06869 (2017)