Why Approximate Matrix Square Root Outperforms Accurate SVD in Global Covariance Pooling?

Yue Song, Nicu Sebe, and Wei Wang

DISI, University of Trento, Trento, Italy
{yue.song nicu.sebe wei.wang}@unitn.it

Abstract

Global Covariance Pooling (GCP) aims at exploiting the second-order statistics of the convolutional feature. Its effectiveness has been demonstrated in boosting the classification performance of Convolutional Neural Networks (CNNs). Singular Value Decomposition (SVD) is used in GCP to compute the matrix square root. However, the approximate matrix square root calculated using Newton-Schulz iteration [14] outperforms the accurate one computed via SVD [15]. We empirically analyze the reason behind the performance gap from the perspectives of data precision and gradient smoothness. Various remedies for computing smooth SVD gradients are investigated. Based on our observation and analyses, a hybrid training protocol is proposed for SVD-based GCP meta-layers such that competitive performances can be achieved against Newton-Schulz iteration. Moreover, we propose a new GCP meta-layer that uses SVD in the forward pass, and Padé approximants in the backward propagation to compute the gradients. The proposed meta-layer has been integrated into different CNN models and achieves state-of-the-art performances on both large-scale and fine-grained datasets.

1. Introduction

Global Covariance Pooling (GCP) explores the second-order statistics by normalizing the covariance matrix of the convolutional features before feeding them to the fully-connected layer. It has been shown to outperform the first-order pooling methods (*e.g.,* max-pooling and average-pooling) [9, 17, 15, 16, 14]. Generally, a GCP meta-layer computes the covariance matrix of the features as the global representation, and then performs eigendecomposition to derive the corresponding eigenvalues and eigenvectors, followed by normalization using either matrix logarithm [9, 17] or the matrix square root [15, 16, 14]. However, the logarithm may change the magnitude of eigenvalues significantly and over-stretch the small ones. Besides, the matrix square root has been proved to amount to robust covariance estimation and to approximately exploit Riemannian geometry [15]. Thus, the matrix square root is often preferred over logarithm normalization.

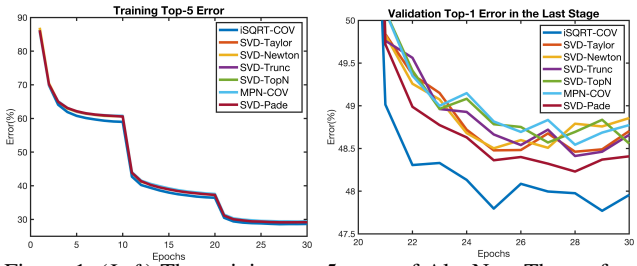

Figure 1. (*Left*) The training top-5 error of AlexNet. The performance gap is gradually minimizing as the learning rate decays. (*Right*) The validation top-1 error in the last stage. These SVD remedies marginally improve the performance but are not comparable against iSQRT-COV [14]. Our proposed SVD-Padé achieves the best results among the SVD methods.

One can either use SVD to compute the accurate square root [15] or use the Newton-Schulz iteration method to derive the approximate square root [14, 16]. Intuitively, the accurate one should yield better performance. Surprisingly, the approximate square root outperforms the exact one continuously [14]. Our paper starts with this intriguing observation and intends to find out the underlying reasons.

One crucial issue of SVD is the numerical instability of its gradients which is derived from the skew-symmetric matrix $\mathbf{K}$ whose off-diagonal elements are defined as $K_{ij}{=}1/(\lambda_i{-}\lambda_j)$, where λ_i and λ_j are eigenvalues. K_{ij} can be reformulated as $\frac{1}{\lambda_i}\frac{1}{1-\lambda_j/\lambda_i}$. When the two eigenvalues are very small and close to each other, $\frac{1}{\lambda_i}$ and $\frac{1}{1-\lambda_j/\lambda_i}$ will move towards infinity. As a consequence, the gradient K_{ij} will explode. To avoid this issue, several attempts have been made to smooth the gradients [16, 7, 25]. These methods consistently outperform ordinary SVD in other tasks, but none of them has been validated in GCP yet.

The gradient instability issue becomes more critical for GCP as it usually deals with very large matrices. According to our observation, when the covariance matrix dimension is very large ($>$200), it is more likely to have many small eigenvalues. Single precision (*i.e.*, float32) usually zeros out small eigenvalues and cannot guarantee the convergence of the network. Therefore, the data type of the covariance is set to double precision (*i.e.*, float64) such that the small eigenvalues can be well represented. However, the high precision can also represent very subtle differences between the small eigenvalues. This can easily result in large gradients and aggravate the instability issue, and thus lead to inferior performance. Therefore, a couple of questions arise:

1) Is the performance gap between MPN-COV [15] (with accurate SVD) and iSQRT-COV [14] (with approximate one) related to their gradient smoothness?
2) Can we smooth the gradient of the accurate SVD to help MPN-COV [15] to achieve competitive performance against iSQRT-COV [14]?

To answer these questions, we introduce several SVD backward remedies into MPN-COV [15] which use different tricks (*e.g.,* gradient truncation, abandoning small eigenvalues, and Taylor polynomial approximation [25]) to smooth the gradients. Fig. 1 shows the training and validation error curves of the modified SVD remedies and the ordinary SVD. We can see that although the modified SVD functions bring marginal performance gain over the ordinary SVD, still none of them is comparable against the Newton-Schulz based iSQRT-COV [14]. This implies that *gradient smoothness does not fully account for the disparity*.

Another interesting observation is that the performance gap between the modified SVD functions and iSQRT-COV [14] is gradually mitigating when the learning rate decreases (see Fig. 1 left). This is probably because with a small learning rate and stable network weights, the covariance matrices are more likely to be well-conditioned, and thus the smallest eigenvalues λ_{min} are larger than EPS (*i.e.*, the smallest positive number the data precision allows). This could benefit SVD for stable eigendecompostion as smaller round-off errors and smoother gradients are obtained. Otherwise, if λ_{min} is smaller than EPS, the small eigenvalues would be zeroed out and the gradient K_{ij} would go to infinity. We empirically show that the covariance matrices of MPN-COV [15] are indeed becoming better-conditioned as the learning rate decreases, which is coherent with the trend of performance gap. This has enlightened us to combine the SVD function with iSQRT-COV [14] and to develop a hybrid training protocol, *i.e.*, use Newton-Schulz iteration to train the network until the learning rate is sufficiently small and the network weights are relatively stable, then switch to the ordinary/modified SVD for accurate matrix square root calculation. By doing so, SVD only deals with well-conditioned matrices in the later stage. This hybrid strategy fully explores the potential of SVD for eigendecomposition, and thus these SVD methods achieve competitive and sometimes better performance than iSQRT-COV [14].

When the hybrid training protocol is used, unlike the situation in the standalone training, the marginal performance improvement of the modified SVD remedies over ordinary SVD no longer holds. The ordinary SVD function can sometimes outperform the modified SVD remedies with smooth gradients. This phenomenon makes us reconsider the effectiveness and necessity to smooth the gradients. As the large gradients can be close to infinity and are very likely to cause overflow, these modified SVD remedies drastically change the gradient at the order of magnitude for smoothness. However, the large but accurate gradients are important for learning robust representation and improving the generalization performance. We argue that **the large gradients should be closely approximated while avoiding singularities**. Among the SVD remedies, Wang *et al.* [25] suggested a promising direction to approximate the gradients using the Taylor polynomial, but their truncated Taylor series cannot converge under certain circumstances. Motivated by this work, we propose to use Padé approximants, a rational approximation technique that has larger convergence radii and more powerful approximation abilities, to estimate the gradients. We show the appealing convergence property of Padé approximants over Taylor polynomial. The proposed meta-layer outperforms all the existing spectral methods and its combination with Newton-Schulz iteration achieves state-of-the-art performances. Our contributions are threefold:

- We empirically analyze the reason behind the superior performance of the approximate matrix square root over the accurate one from data precision and gradient smoothness view points. Various remedies for computing smooth SVD gradients are investigated and validated in GCP.
- A hybrid training protocol is proposed for GCP meta-layers and competitive performance can be achieved compared with Newton-Schulz iteration. We justified this strategy using the metric condition number to measure the ill-condition of covariance matrices.
- We propose a SVD backward algorithm that relies on Padé approximants for fast and robust gradient approximation. It consistently achieves state-of-the-art performance on different datasets and different models.

Finally, to promote the easy applicability of the relevant SVD techniques, we will release the source codes of all the methods implemented in PYTORCH upon acceptance[1].

[1]The code is available at https://github.com/KingJamesSong/DifferentiableSVD.

2. Related Work

2.1. Differentiable SVD

As a traditional matrix decomposition technique, SVD has a wide range of applications in modern deep learning, including batch whitening [25, 8], style transfer [2, 3], and image segmentation [1, 9]. The theory of differentiable SVD was first formulated in [9, 10]. When two eigenvalues are very close, the spurious gradient explosion is likely to happen and cause numerical instability. To circumvent this issue, [7] proposed to divide the matrix into two smaller sub-matrices, which reduces the risk of having small and close eigenvalues. Nonetheless, this modification has no theoretical basis and may lead to inferior performances. [25] proposed to rely on power iteration to iteratively calculate the approximate gradient. However, power iteration converges only if the largest eigenvalue λ_1 is dominant and this requirement may limit its practical usage (*i.e.*, the top two largest values may be equal). [25] proposed to use Taylor expansion for SVD gradient estimation. Due to the singularity of the function to approximate, the Taylor polynomial cannot give a good approximation when it is close to the polar singularity. Motivated by [25], we propose to use Padé approximants, a rational approximation technique to compute the SVD gradients. Compared with the Taylor polynomial, Padé approximants have larger convergence radii and more powerful approximation abilities.

2.2. Global Covariance Pooling

In deep neural networks, the global covariance pooling layer targets exploring the second-order statistics of the high-level representations before the fully-connected layer. Its powerful ability in boosting the network performance has been demonstrated in the past years [9, 17, 22, 16, 15, 14, 21, 23, 24]. DeepO^{2}P [9] was the first end-to-end global covariance pooling network. It relies on SVD to compute the covariance and then conducts matrix logarithm for nonlinear normalization. B-CNN [17] was proposed to aggregate the outer product of convolutional features from two networks and then performs element-wise power normalization. Improved B-CNN [16] investigated various ways of matrix normalization techniques and proved that the matrix square root normalization significantly outperforms other normalization schemes. Besides, they suggested alternate matrix square root computation techniques: using the Newton-Schulz iteration in the forward pass and solving the Lyapunov equation for gradient computation during backpropagation. G^{2}DeNet [22] inserted a Gaussian distribution into the network and considered the geometry of the Gaussian manifold. MPN-COV [15] presented a matrix power normalization method for robust covariance estimation. For a GPU-friendly computation concern, iSQRT-COV [14] proposed to use the Newton-Schulz iteration to accelerate the approximate matrix square root computation during both forward and backward calculations. Our paper conducts an investigation into the performance gap between [15] and [14]. The former method computes the exact matrix square root, while the latter calculates the approximate one but achieves better performance.

3. Investigation

3.1. Matrix Square Root Revisited

Given the representation $\mathbf{X}\in\mathbb{R}^{d\times N}$ before the fully-connected layer of a neural network where d defines the dimensionality and N denotes the number of features, both accurate and approximate algorithms [15, 14] calculate the matrix square root of its sample covariance to exploit the second-order statistics.

Accurate Sqaure Root: MPN-COV [15]. After the convolutional layers, the extracted representation $\mathbf{X}\in\mathbb{R}^{d\times N}$ is used to compute the covariance matrix:

$$\mathbf{P}=\mathbf{X}\bar{\mathbf{I}}\mathbf{X}^T \tag{1}$$

where $\bar{\mathbf{I}}=\frac{1}{N}(\mathbf{I}-\frac{1}{N}\mathbf{1}\mathbf{1}^T)$ represents the centering matrix, $\mathbf{I}$ denotes the identity matrix, and $\mathbf{1}$ is a column vector whose values are all ones, respectively. After centralization, the covariance matrix $\mathbf{P}$ is symmetric positive semi-definite. The eigendecomposition can be performed via either SVD or EIG:

$$\mathbf{P}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{2}$$

here $\mathbf{\Lambda}=\mathrm{diag}(\lambda_1,\dots,\lambda_\mathrm{d})$ is the diagonal matrix with eigenvalues arranged in a non-increasing order, and $\mathbf{U}=[\mathbf{u}_1,\dots,\mathbf{u}_d]$ is an orthogonal matrix where each column $\mathbf{u}_i$ is the eigenvector that corresponds to the eigenvalue λ_i. The matrix square root is obtained via the following equation:

$$\mathbf{Q}\triangleq\mathbf{P}^{\frac{1}{2}}=\mathbf{U}\mathbf{F}(\mathbf{\Lambda})\mathbf{U}^T \tag{3}$$

where $\mathbf{F}(\lambda)$ is the diagonal matrix $\mathrm{diag}(\lambda_1^{\frac{1}{2}},\dots,\lambda_\mathrm{d}^{\frac{1}{2}})$. Afterwards, the global representation $\mathbf{Q}$ is fed into the fully-connected layer. During back-propagation, the partial derivative of loss function l w.r.t. the input matrix $\mathbf{X}$ is computed based on the matrix backpropagation methodology [9, 10]. Let $\frac{\partial l}{\partial\mathbf{Q}}$ denote the gradient from the last fully-connected layer, we can derive the partial derivative of eigenvector matrix $\mathbf{U}$ and eigenvalue matrix $\mathbf{\Lambda}$ as:

$$\begin{aligned}\frac{\partial l}{\partial\mathbf{U}}&=(\frac{\partial l}{\partial\mathbf{Q}}+(\frac{\partial l}{\partial\mathbf{Q}})^T)\mathbf{U}\mathbf{F},\\ \frac{\partial l}{\partial\mathbf{\Lambda}}&=\frac{1}{2}(\mathrm{diag}(\lambda_1^{-\frac{1}{2}},\dots,\lambda_\mathrm{d}^{-\frac{1}{2}})\mathbf{U}^\mathrm{T}\frac{\partial l}{\partial\mathbf{Q}}\mathbf{U})_\mathrm{diag}\end{aligned} \tag{4}$$

where $(\cdot)_\mathrm{diag}$ denotes diagonalization that keeps only the diagonal elements of the matrix. Subsequently, the derivative of the covariance $\mathbf{P}$ can be calculated using the chain rule:

$$\frac{\partial l}{\partial\mathbf{P}}=\mathbf{U}((\mathbf{K}^T\circ(\mathbf{U}^T\frac{\partial l}{\partial\mathbf{U}}))+(\frac{\partial l}{\partial\mathbf{\Lambda}})_\mathrm{diag})\mathbf{U}^T \tag{5}$$

where $\circ$ represents matrix Hadamard product, and the matrix $\mathbf{K}$ is comprised of elements $K_{ij}=1/(\lambda_i-\lambda_j)$ if $i\neq j$

and K_{ij}=0 otherwise. The loss l w.r.t. the input feature $\mathbf{X}$ is finally computed as:

$$\frac{\partial l}{\partial \mathbf{X}} = (\frac{\partial l}{\partial \mathbf{P}} + (\frac{\partial l}{\partial \mathbf{P}})^T)\mathbf{X}\bar{\mathbf{I}} \tag{6}$$

The whole network can be trained end-to-end using the forward and backward pass defined in eqs. (1) to (6). For the detailed loss derivations, the reader is kindly referred to [9, 15] for a comprehensive review.

Approximate Square Root: iSQRT-COV [14]. For the concern of GPU-friendly computation efficiency, [14] proposed a loop-embedded meta-layer to iteratively compute the approximate matrix square root via Newton-Schulz iteration [6]. Specifically, for computing the square root $\mathbf{Y}$ of a matrix $\mathbf{A}$, the coupled Newton-Schulz iteration takes the following form:

$$\begin{aligned} \mathbf{Y}_k &= \frac{1}{2}\mathbf{Y}_{k-1}(3\mathbf{I} - \mathbf{Z}_{k-1}\mathbf{Y}_{k-1}), \\ \mathbf{Z}_k &= \frac{1}{2}(3\mathbf{I} - \mathbf{Z}_{k-1}\mathbf{Y}_{k-1})\mathbf{Z}_{k-1} \end{aligned} \tag{7}$$

where $\mathbf{Y}_k$ is initialized with $\mathbf{Y}_0$=$\mathbf{A}$, and $\mathbf{Z}_k$ starts with $\mathbf{Z}_0$=$\mathbf{I}$. Since Newton-Schulz iteration converges locally only if $\|\mathbf{A}-\mathbf{I}\|<1$, the covariance matrix $\mathbf{P}$ is first pre-normalized by its trace to meet the convergence condition:

$$\mathbf{A} = \frac{1}{\mathrm{tr}(\mathbf{P})}\mathbf{P} \tag{8}$$

Next, the normalized matrix $\mathbf{A}$ takes Newton-Schulz iteration and outputs the approximate matrix square root $\mathbf{Y}_N$ after N iterations. The pre-normalization in eq. (8) non-trivially changes the data magnitude, which may cause the network to fail to converge. After Newton-Schulz iteration, the resultant matrix $\mathbf{Y}_N$ is post-compensated as follows:

$$\mathbf{Q} = \sqrt{\mathrm{tr}(\mathbf{P})}\mathbf{Y}_N \tag{9}$$

The back-propagation algorithm for iSQRT-COV [14] is not that straightforward as MPN-COV [15]. The loss partial derivative $\frac{\partial l}{\partial \mathbf{Y}_k}$ and $\frac{\partial l}{\partial \mathbf{Z}_k}$ need to be calculated for each loop of Newton-Schulz iteration. For conciseness, we only give the derivative with the covariance here:

$$\begin{aligned} \frac{\partial l}{\partial \mathbf{P}} = &-\frac{1}{(\mathrm{tr}(\mathbf{P}))^2}\mathrm{tr}((\frac{\partial l}{\partial \mathbf{A}})^T\mathbf{P})\mathbf{I} + \frac{1}{\mathrm{tr}(\mathbf{P})}\frac{\partial l}{\partial \mathbf{A}} \\ &+\frac{1}{2\sqrt{\mathrm{tr}(\mathbf{P})}}\mathrm{tr}((\frac{\partial l}{\partial \mathbf{Q}})^T\mathbf{Y}_N)\mathbf{I} \end{aligned} \tag{10}$$

Then eq. (6) can be used again to derive the loss derivative with regards to the input feature.

3.2. Data Precision Analysis

As discussed before, GCP methods require double precision to ensure the effective numerical representation for small eigenvalues. To validate the impact of the data precision (*i.e.,* single or double), we take AlexNet [13] as the backbone and conduct experiments using both meta-layers in different data precision on ImageNet [4]. AlexNet is chosen because it is light-weighted and easy to extrapolate to other deep models. The left of Fig. 2 presents the training error curve of the meta-layers. The data precision has a slight impact on iSQRT-COV [14] (about 0.1%), whereas MPN-COV [15] can be influenced substantially. MPN-COV [15] can achieve reasonable performances in double precision but fails to converge using single precision. This observation confirmed the necessity of high precision to allow for effective numerical representation of eigenvalues.

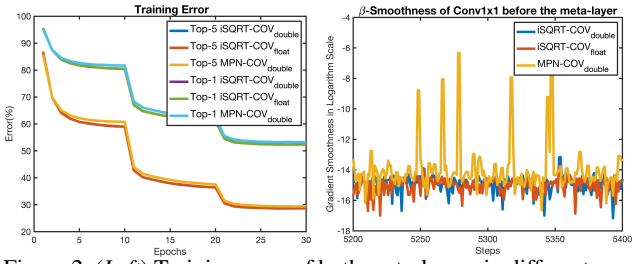

Figure 2. (*Left*) Training error of both meta-layers in different precision. (*Right*) The effective β-smoothness [18] of the two methods. Larger value indicates less smooth gradients.

From eq. (5), we can see that the small eigenvalues with high precision can easily result in large K_{ij} and cause gradient overflow. We measure the effective β-smoothness [18] of both meta-layers to quantify their gradient smoothness. As can be observed in Fig. 2 right, MPN-COV [15] has far less smooth gradients than iSQRT-COV [14]. To attain similar gradient smoothness for SVD, we explore different approaches to manipulate the backward algorithm of SVD for obtaining smooth gradients. In the rest of the paper, unless explicitly specified, we keep using double precision for GCP meta-layer of all the methods.

3.3. Gradient Smoothness Analysis

To investigate the concrete impact of gradient smoothness, we design several SVD meta-layers with smooth gradient. These meta-layers all use SVD as the forward pass but have different configurations during back-propagation.

Top-N Eigenvalue. The first approach is to directly abandon small eigenvalues that are likely to trigger numerical instability from the diagonal matrix $\mathbf{\Lambda}$. Since the diagonal eigenvalues are sorted in descending order, we can simply choose the top N eigenvalues and discard the rest. The modification can be formally formulated as:

$$\hat{\mathbf{\Lambda}} = \mathrm{diag}(\lambda_1, \dots, \lambda_N, 0, \dots) \tag{11}$$

The number of kept eigenvalues N is chosen through cross-validation. This method is denoted as "SVD-TopN".

Gradient Truncation. Our second method is to limit the magnitude of gradient and apply truncation on matrix $\mathbf{K}$ during back-propagation. Specifically, we have

$$\hat{K}_{ij} = \begin{cases} \mathrm{T}, & \text{if } \frac{1}{\lambda_i - \lambda_j} > \mathrm{T} \\ -\mathrm{T}, & \text{if } \frac{1}{\lambda_i - \lambda_j} < -\mathrm{T} \end{cases} \tag{12}$$

where T is a large constant under the precision of the data type, and its optimal value can also be set by cross-validation. We name this approach "SVD-Trunc".

Power Iteration Gradient. One recent SVD backward al-

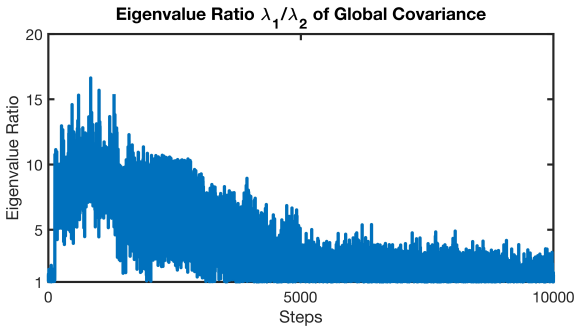

Figure 3. The ratio of the first two eigenvalues (λ_1/λ_2). The first eigenvalue is not dominant for every covariance matrix.

gorithm suggested using power iteration to compute the associated gradients [25]. Formally, power iteration takes the iterative update $\mathbf{u}^k{=}\mathbf{P}\mathbf{u}^{k-1}/||\mathbf{P}\mathbf{u}^{k-1}||$ to approximate the eigenvectors. By relying on $||\mathbf{P}\mathbf{u}_1||{=}||\lambda_1\mathbf{u}_1||$ when λ_1 is dominant, the gradient in eq. (5) can be re-written as:

$$\frac{\partial l}{\partial \mathbf{P}} = \mathbf{u}_1((\mathbf{K}^T \circ (\mathbf{u}_1^T \frac{\partial l}{\partial \mathbf{u}_1})) + (\frac{\partial l}{\partial \mathbf{\Lambda}})_{\text{diag}})\mathbf{u}_1^T \tag{13}$$

However, the convergence of power method requires that the first eigenvalue is dominant ($\lambda_1/\lambda_2{>}1$). We empirically observe that this method fails to converge in GCP. Fig. 3 displays the ratio of first two eigenvalues in the first $10,000$ training steps. As the network training goes, the ratio is gradually decreasing and reaches 1 for some covariance matrices. We thus conjecture that it cannot converge because the first eigenvalue λ_1 is not always dominant.

Newton-Schulz Gradient. Our third remedy is to use Newton-Schulz iteration formulated in eq. (7) for back-propagation while using SVD as the forward pass. The iteration times are carefully tuned to achieve the best performances. This method is denoted as "SVD-Newton".

Taylor Polynomial Gradient. Recently, [25] proposed to use Taylor expansion to approximate the SVD backward gradients. They re-formulated the non-zero elements of the matrix $\mathbf{K}$ computed in eq. (5) as a composition:

$$K_{ij} = \frac{1}{\lambda_i - \lambda_j} = \frac{1}{\lambda_i} \cdot \frac{1}{1-(\lambda_j/\lambda_i)} \tag{14}$$

Notice that the right term resembles the function $f(x){=}1/(1{-}x)$. The Maclaurin series at $x{=}0$ of its Taylor expansion can be expressed as:

$$P(z) = \sum_{i=0}^{K} z^i + R(z^{K+1}) \tag{15}$$

where $\sum_{n=0}^{K} z^n$ represents the Taylor expansion to degree K, and $R(z^{K+1})$ denotes the discarded remainder of higher degree. Injecting eq. (15) into eq. (14) leads to:

$$K_{ij} \approx \frac{1}{\lambda_i}(1+\frac{\lambda_j}{\lambda_i}+(\frac{\lambda_j}{\lambda_i})^2+\cdots+(\frac{\lambda_j}{\lambda_i})^K) \leq \frac{K+1}{\lambda_i} \tag{16}$$

Now the numerical infinity $1/(\lambda_i{-}\lambda_j)$ when the two eigenvalues are close has disappeared in the equation, and we have a bounded gradient estimation. From Cauchy root test, the Taylor series in eq. (15) only converge in the range $-1{<}z{<}1$. In the case of $j{<}i$, the ratio λ_j/λ_i is larger than 1 and thus outsides the convergence radius. To avoid this issue, the matrix $\mathbf{K}$ is split into two triangular sub-matrix where the upper triangle defines the cases $j{>}i$ and the lower triangle consists of elements when $j{<}i$. As $\mathbf{K}$ is a skew-symmetric matrix, only the upper part needs to be computed. We call this method "SVD-Taylor".

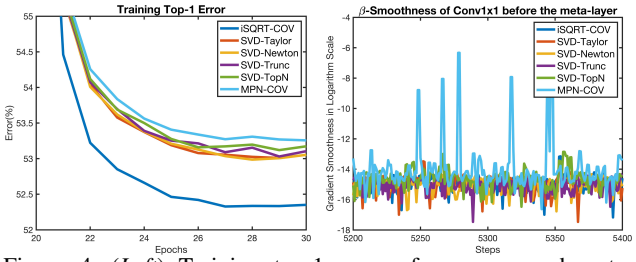

Figure 4. (*Left*) Training top-1 error of our proposed meta-layers. These methods bring about maximally 0.2% performance improvements over MPN-COV [15]. (*Right*) The effective β-smoothness [18] of the these methods. The gradient is smoothed to the similar degree with iSQRT-COV [14].

We integrate the aforementioned methods into AlexNet [13] and evaluate their performances on ImageNet [4]. Fig. 4 compares the training top-1 error of these methods (left) and their effective β-smoothness [18] (right). The proposed SVD variants could smooth the SVD gradients to different extents and improve the performances by maximally 0.2%. However, iSQRT-COV [14] still leads these SVD remedies by a large margin.

3.4. Hybrid Training Protocol

Though these SVD remedies can not outperform iSQRT-COV [14], their performance gap keeps decreasing as the learning rate decays. This phenomenon elicits our guess that the covariance matrices might be better-conditioned for SVD when the learning rate is sufficiently small and the model weights are well-trained. The hypothesis could be validated using condition number, *i.e.*, a metric to quantify the ill-condition of covariance matrices. Condition number is defined by the ratio of the eigenvalues $\lambda_{max}/\lambda_{min}$ and can measure the stability of matrices. A high condition number indicates that λ_{min} is relatively small and the matrix is close to singular, while a low condition number assures that λ_{min} is relatively large and the matrix is well-conditioned. As a rule of thumb, when using double precision, matrices with condition numbers greater than $1e14$ are considered unstable and ill-conditioned. Ill-conditioned matrices are more likely to have eigenvalues that are smaller than EPS. These small eigenvalues would be zeroed out and the gradient K_{ij} might move to infinity, which could trigger round-off error and gradient explosion. We measure the average condition number of MPN-COV [15] of AlexNet during different training stages. Table 1 shows the evaluation results of three epochs with different learning rates. As the training goes, the condition number gradually decreases. In the first stage, the average condition number

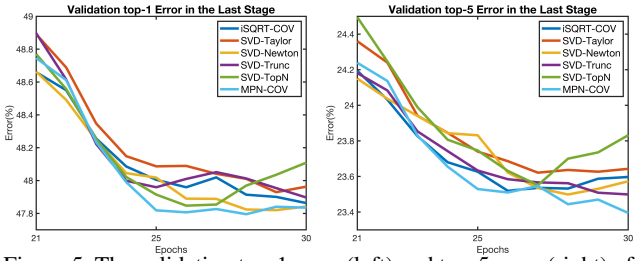

Figure 5. The validation top-1 error (left) and top-5 error (right) of AlexNet in the last stage using the hybrid training strategy. All the lines are smoothed by moving average filter for better view.

is higher than the ill-condition threshold $1e14$. But in the later stages, MPN-COV [15] shows much smaller average condition number and therefore has better-conditioned covariance matrices. This could benefit SVD for more stable eigendecomposition.

Table 1. Average condition number of different epochs. Higher value indicates the matrix is less stable and closer to singular.

Methods	Epoch 1	Epoch 11	Epoch 21
MPN-COV [15]	3.41 e14	5.36 e13	1.70 e13

This finding motivates us to design a hybrid training strategy such that SVD only needs to deal with well-conditioned matrices in later stages. In general, a deep model trained on ImageNet [4] usually starts with a large learning rate and decays the rate twice when the error plateaus. We propose to use Newton-Schulz iteration to train the model before the last learning rate decay. Then the model is switched to SVD meta-layer to warm up the weights for few epochs. Finally, we decay the learning rate and keep using SVD meta-layer to tune the network till the end. Fig. 5 shows the validation error of AlexNet using the proposed hybrid training strategy in the last stage. All the SVD variants have achieved competitive and even slightly better performances than iSQRT-COV [14]. Unlike the case of standalone training, when hybrid training strategy is applied, these SVD remedies with smooth gradients do not show obvious improvement over ordinary SVD. This observation challenges the necessity and effectiveness to smooth the gradients.

4. Differentiable SVD by Padé Approximants

Based on our above observation and analysis, we hazard that **the backward gradients should be closely approximated on the premise that the singularity is avoided**. However, existing methods sacrifice the large gradients for smoothness and none of them well approximate the gradients close to the singularities. Motivated by Taylor polynomial estimation [25], we propose a more robust gradient approximation method for differentiable SVD.

4.1. Taylor Polynomial Problem.

Although Taylor polynomial provides a promising direction for gradient estimation, there still exist some caveats. First, $f(x)=1/(1-x)$ is a meromorphic function and has a pole at $x=1$. Its corresponding geometric series $P(z)=\sum_i^\infty z^i$ converge only when $|z|<1$. In the scope of approximation theory [19], meromorphic functions that contain poles may not be well approximated by polynomial approximation techniques (*e.g.*, Taylor and Chebyshev polynomial), as polynomials are entire functions without singularities. The closer the distance to the polar singularity ($x=1$) is, the worse its approximation ability would be. When the two eigenvalues are very close ($\lambda_i/\lambda_j \approx 1$), the Taylor polynomial would move to the convergence boundary and reach the upper bound $K_{ij}=(K+1)/\lambda_i$. This would result in a poor approximation and make the Taylor polynomial sensitive to its truncation degree K. Rational approximation, on the other hand, usually gives a better approximation for function with poles because the use of rational functions allows singularities to be well represented. To address the aforementioned issues, we propose to substitute the truncated Taylor series with Padé approximants, a kind of rational approximation that has enlarged convergence radii for more robust gradient estimation. In the next sections, we will introduce the definition of Padé approximants, the formulation in the SVD algorithm, and their appealing convergence property over Taylor polynomial.

4.2. Padé Approximants

The Padé approximants are a particular type of rational fraction approximation. The idea is use the ratio of two polynomials to match a power series. Consider a Maclaurin Taylor series $A(x)=\sum_{i=0}^\infty a_i x^i$ to match, we have the corresponding formal definition of Padé approximants:

$$[M/N] = \frac{P_M(x)}{Q_N(x)} = \frac{\sum_{m=0}^{M} p_m x^m}{1+\sum_{n=1}^{N} q_n x^n} \tag{17}$$

where the numerator $P_M(x)$ is a polynomial of degree at most M, and the denominator $Q_N(x)$ is a polynomial of degree at most N. A $[M/N]$ Padé approximants agree up to a Taylor series of order $M+N+1$. Their coefficients are determined by matching the power series:

$$A(x) - \frac{P_M(x)}{Q_N(x)} = R(x^{M+N+1}) \tag{18}$$

By introducing eq. (17) into this equation and setting the remainder $R(x^{M+N+1})$ to zero, we can linearize the coefficient equations as:

$$\begin{aligned} a_0 &= p_0 \\ a_1 + a_0 q_1 &= p_1 \\ &\vdots \\ a_M + a_{M-1} q_1 + \cdots + a_0 q_M &= p_M \\ a_{M+1} + a_M q_1 + \cdots + a_{M-N+1} q_N &= 0 \\ &\vdots \\ a_{M+N+1} + a_{M+N-1} q_1 + \cdots + a_M q_N &= 0 \end{aligned} \tag{19}$$

where $a_n \equiv 0$ for $n<0$ and $q_j \equiv 0$ for $j>M$. Solving these equations directly gives the Padé coefficients.

4.3. Formulation in SVD Back-propagation

Let us start with the gradient approximated by Taylor polynomial. The core idea of Taylor gradient is to use a truncated Taylor series in eq. (15) of degree K to approximate the elements of matrix $\mathbf{K}$ defined in eq. (14). The Padé approximants can be derived by matching eq. (15):

$$\frac{P_M(x)}{Q_N(x)} + R(x^{M+N+1}) = \sum_{i=0}^{K} x^i + R(x^{K+1}) \tag{20}$$

Then we can get the matched Padé approximants and re-express the non-zero matrix elements of $\mathbf{K}$ as

$$K_{ij} \approx \frac{1}{\lambda_i} \cdot \frac{P_M(\lambda_j/\lambda_i)}{Q_N(\lambda_j/\lambda_i)} = \frac{1}{\lambda_i} \cdot \frac{\sum_{m=0}^{M} p_m (\frac{\lambda_j}{\lambda_i})^m}{1 + \sum_{n=1}^{N} q_n (\frac{\lambda_j}{\lambda_i})^n} \tag{21}$$

In practice, diagonal $[M/N]$ Padé approximants where the numerator and denominator have the same degree ($M=N+1$) are often preferred. This will guarantee stable Padé coefficients and the enlarged convergence range.

4.4. Enlarged Convergence Range.

Padé approximants are derived out of a finite Taylor series, but they have more desired convergence properties than truncated Taylor series. Specifically for diagonal Padé approximants, we have:

Theorem 1. *If the function $f(z)$ is a Stieltjes transform $f(z)=\int_a^b \frac{1}{z-x} d\mu(x)$ of a compactly supported measure $\mu(x)$ in $[a,b]$, then the associated $[N+1/N]$ diagonal Padé approximants are orthogonal and there exists such function $r(x)>1$ that the convergence $\lim_{N\to\infty} |f(z) - \frac{P_{N+1}(z)}{Q_N(z)}|^{\frac{1}{N}} = \frac{1}{r^2}$ is exponential in $[a,b]$.*

The convergence theorem along with the asymptotic behavior of diagonal Padé approximants were given in [20]. Notice that the function $f(x)=1/(1-x)$ is compactly supported in $\mathbb{R}$, as it only vanishes at infinity. Thus, this property can ensure that the associated Padé approximants still have good approximations even close to the convergence boundary of the original Taylor series. That being said, a diagonal Padé approximant can not only avoid singularities of the SVD gradients but also give a very close approximation for any possible eigenvalue ratio λ_j/λ_i.

5. Experiment

5.1. Models and Datasets

Following [15, 14], we first take AlexNet [13] and ResNet-50 [5] as the backbones and conduct experiments on ImageNet 2012 [4] for the large-scale visual recognition. After training GCP models on ImageNet, we evaluate the methods on Fine-Grained Visual Categorization (FGVC). The pre-trained ResNet-50 with different GCP meta-layers using hybrid training strategy are fine-tuned on three popular fine-grained benchmarks, *i.e.,* Caltech Birds (Birds) [26], Stanford Dogs (Dogs) [11], and Stanford Cars (Cars) [12]. More details about the datasets and implementation can be found in the Appendix.

Table 2. Validation error of AlexNet using different training strategy. The best three results are highlighted in red, blue, and green.

Methods	Standalone Training		Hybrid Training			
	Final Error (%)		Final Error (%)		Best Error (%)	
	top-1	top-5	top-1	top-5	top-1	top-5
iSQRT-COV [14]	47.95	23.64	47.95	23.64	47.81	23.54
SVD-Padé	48.41	23.91	47.76	23.48	47.63	23.21
SVD-Taylor	48.70	24.30	47.92	23.56	47.86	23.56
SVD-Newton	48.86	25.08	47.87	23.48	47.77	23.38
SVD-Trunc	48.66	24.10	47.98	23.45	47.81	23.48
SVD-TopN	48.56	24.10	47.96	23.65	47.81	23.54
MPN-COV [15]	48.77	24.28	47.94	23.54	47.75	23.24

5.2. Results on ImageNet with AlexNet

Table 2 displays the validation errors of AlexNet using the standalone and hybrid training strategies. In the case of standalone training, the proposed SVD-Padé method outperforms other SVD remedies significantly and improves the ordinary SVD algorithm by about 0.4% in both top-1 and top-5 error. Since the performance gap between each method is subtle and may fluctuate when using the hybrid training strategy, we report both the final and best validation error for comprehensive comparisons. When hybrid training protocol is used, our proposed SVD-Padé achieves state-of-the-art performances in four metrics and surpasses all the other baselines including iSQRT-COV [14].

5.3. Results on ImageNet with ResNet

Table 3. Validation errors of ResNet-50 and ResNet-101. The best three results are highlighted in red, blue, and green.

	Methods	Standalone Training		Hybrid Training	
		top-1	top-5	top-1	top-5
ResNet-50	iSQRT-COV [14]	22.81	6.60	22.81	6.60
	SVD-Padé	22.67	6.51	22.60	6.44
	SVD-Taylor	22.91	6.67	22.77	6.53
	SVD-Newton	22.86	6.65	22.72	6.55
	SVD-Trunc	22.85	6.70	22.74	6.52
	SVD-TopN	22.91	6.68	22.76	6.51
	MPN-COV [15]	22.93	6.75	22.79	6.50
	Vanilla ResNet-50	23.85	7.13	23.85	7.13
ResNet-101	iSQRT-COV [14]	21.60	5.88	21.60	5.88
	SVD-Padé	21.48	5.80	21.40	5.69
	MPN-COV [15]	21.79	5.99	21.58	5.80
	Vanilla ResNet-101	22.63	6.44	22.63	6.44

Tables 3 compares the validation errors of ResNet using different training strategies. The results are very coherent with those of AlexNet. Our SVD-Padé achieves the best evaluation results. Even when the standalone training protocol is applied, *i.e.,* the SVD methods are trained from scratch, the SVD-Padé can still outperform [14] by about 0.2 %. This demonstrates the necessity of closely approximating gradients. When it comes to the hybrid training strategy, these SVD variants have achieved competitive performances against iSQRT-COV [14] and even outperform by a narrow margin. In particular, our SVD-Padé remedy

Table 4. Comparison of accuracy (%) on fine-grained datasets using ResNet-50 with different GCP meta-layers as backbone. The best three results are highlighted in red, blue, and green respectively. / means the method cannot converge.

Methods	Birds [26]		Dogs [11]		Cars [12]	
	Final	Best	Final	Best	Final	Best
iSQRT-COV [14]	85.95	86.45	82.34	83.45	90.93	91.56
SVD-Padé	87.05	87.29	83.40	84.34	92.55	92.99
SVD-Taylor	86.95	87.20	83.23	84.22	92.46	92.71
SVD-Newton	86.97	87.22	83.08	83.94	92.35	92.51
SVD-Trunc	87.16	87.25	82.95	84.03	92.43	93.04
SVD-TopN	/	/	/	/	/	/
MPN-COV [15]	86.89	87.19	83.34	84.24	92.45	92.84

also achieves the best results. We also believe that warming up more epochs when switching to SVD methods can a bring larger performance gain.

5.4. Results on FGVC with ResNet

We finally validate the performances of these GCP meta-layers on FGVC. The ResNet-50 models with different GCP meta-layers are first pre-trained on ImageNet using the proposed hybrid training strategy and then fine-tuned on each FGVC dataset. Table 4 compares their best and final validation accuracy. As can be observed, all the SVD variants have very competitive performances and surpass Newton-Schulz iteration by a large margin with 0.6%, although some of them slightly trail Newton-Schulz iteration on ImageNet. This demonstrates our guess that accurate matrix square root also works better on FGVC datasets with small learning rate and well-trained network weights. Furthermore, the large margin may imply that the GCP networks trained by precise square root have better generalization capabilities than the approximate ones. Among these SVD variants, our proposed SVD-Padé method achieves the best performance and outperforms Newton-Schulz iteration by 1% across datasets. Besides, the SVD-TopN method cannot converge on any fine-grained datasets. This meets our expectation as the small eigenvalues may encode the class-specific features of the fine-grained classes and thus play an important role in FGVC. Despite the preliminary analysis, the problem is worth further investigation.

5.5. Speed Comparison

Table 5. Computation time cost of each GCP meta-layer for a single batch on AlexNet and the computation bottleneck.

Methods	Bottleneck	FP (s)	BP (s)	Total (s)
iSQRT-COV	N/A	0.23	0.53	0.76
SVD-Padé	SVD	0.96	0.28	1.24
SVD-Taylor		0.96	0.32	1.28
SVD-Newton		2.36	1.05	3.41
SVD-Trunc		0.97	0.26	1.23
SVD-TopN		0.98	0.24	1.22
MPN-COV		0.97	0.23	1.20

We compare the time cost of a single batch for each meta-layer on AlexNet in Table 5. Compared with iSQRT-COV [14], our implementation consumes less back-propagation time as iterative matrix-matrix multiplication is not involved in the SVD backward algorithm. The computational bottleneck of our implementation is mainly the forward eigendecomposion, which is unfortunately limited by the platform. Our SVD-Padé has faster backward speeds than SVD-Taylor, given that they agree up to the Taylor series of the same degree K. This is mainly because the numerator and denominator of Padé approximants can be computed in parallel, the total iteration times would be $K/2$. For SVD-Taylor, it would take K iterations to reach the same degree.

5.6. Approximation Error of Taylor and Padé

Table 6. Approximation error of Taylor polynomial and diagonal Padé approximants of degree 100 in double precision.

λ_j/λ_i	0.1	0.3	0.5	0.7	0.9	0.99	0.999
Taylor	9e-19	7e-18	2e-21	8e-16	2e-4	36	904
Padé	9e-19	5e-18	1e-21	5e-17	3e-16	8e-13	3e-10

Table. 6 compares the approximation error to function $f(x)=1/(1-x)$ of both Taylor polynomial and diagonal Padé approximants of degree 100. As discussed in Sec. 4.1, the approximation error of Taylor polynomial gets amplified when the eigenvalue ratio is close to the convergence boundary ($\lambda_j/\lambda_i \approx 1$). By contrast, our Padé approximants consistently provide good approximation for any λ_j/λ_i.

5.7. Upper Bound of Gradient

Table 7. Upper bound of the gradient K_{ij} for each SVD method.

Methods	SVD-Padé	SVD-Taylor	SVD-Trunc	SVD-TopN	SVD-Newton	SVD
Analytical Form	$\frac{1}{\lambda_i} \cdot \frac{\sum_{m=0}^{M} p_m}{1+\sum_{n=1}^{N} q_n}$	$\frac{K+1}{\lambda_i}$	T	$\frac{1}{\lambda_N}$	/	$\frac{1}{\lambda_i-\lambda_j}$
Maximal Value	6.00e36	4.55e17	1e10	4.50e15	/	∞
Trigger Condition	$\lambda_i = \lambda_j \leq$ EPS	$\lambda_i = \lambda_j \leq$ EPS	$\left\|\frac{1}{\lambda_i-\lambda_j}\right\| \geq$ T	$\lambda_N \leq$ EPS	/	$\lambda_i = \lambda_j$

Table. 7 summarizes the upper bound of gradient K_{ij} for each SVD method. Our proposed SVD-Padé allows for the largest upper bound of gradient, but the maximal value is still acceptable in the double precision ($<1.79e308$) or even single precision ($<3.40e38$). For the detailed derivation of upper bound, please refer to the Appendix.

6. Conclusion

In this paper, we empirically analyze why approximate matrix square root outperforms exact ones on GCP from data precision and gradient smoothness. We investigate various SVD remedies with smooth gradients and validate their performances on GCP. We suggest a hybrid training strategy to help the SVD methods to achieve competitive performances against Newton-Schulz iteration. Based on the findings, we propose a new GCP meta-layer that uses SVD as the forward pass and Padé approximants during back-propagation for robust gradient approximation. The proposed meta-layer has achieved state-of-the-art performances on different datasets and deep models.

Acknowledgments

This work was supported by EU H2020 SPRING No. 871245 and EU H2020 AI4Media No. 951911 projects.

References

[1] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 3

[2] Tai-Yin Chiu. Understanding generalized whitening and coloring transform for universal style transfer. In *ICCV*, 2019. 3

[3] Wonwoong Cho, Sungha Choi, David Keetae Park, Inkyu Shin, and Jaegul Choo. Image-to-image translation via group-wise deep whitening-and-coloring transformation. In *CVPR*, 2019. 3

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 4, 5, 6, 7

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7

[6] Nicholas J Higham. *Functions of matrices: theory and computation*. SIAM, 2008. 4

[7] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *CVPR*, 2018. 1, 3

[8] Lei Huang, Lei Zhao, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. An investigation into the stochasticity of batch whitening. In *CVPR*, 2020. 3

[9] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *ICCV*, 2015. 1, 3, 4

[10] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Training deep networks with structured layers by matrix backpropagation. *arXiv preprint arXiv:1509.07838*, 2015. 3

[11] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 7, 8

[12] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 7, 8

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 4, 5, 7

[14] Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *CVPR*, 2018. 1, 2, 3, 4, 5, 6, 7, 8

[15] Peihua Li, Jiangtao Xie, Qilong Wang, and Wangmeng Zuo. Is second-order information helpful for large-scale visual recognition? In *ICCV*, 2017. 1, 2, 3, 4, 5, 6, 7, 8

[16] Tsung-Yu Lin and Subhransu Maji. Improved bilinear pooling with cnns. *arXiv preprint arXiv:1707.06772*, 2017. 1, 3

[17] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015. 1, 3

[18] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003. 4, 5

[19] Michael James David Powell et al. *Approximation theory and methods*. Cambridge university press, 1981. 6

[20] Walter Van Assche. Padé and hermite-padé approximation and orthogonality. *arXiv preprint math/0609094*, 2006. 7

[21] Qilong Wang, Peihua Li, Qinghua Hu, Pengfei Zhu, and Wangmeng Zuo. Deep global generalized gaussian networks. In *CVPR*, 2019. 3

[22] Qilong Wang, Peihua Li, and Lei Zhang. G2denet: Global gaussian distribution embedding network and its application to visual recognition. In *CVPR*, 2017. 3

[23] Qilong Wang, Jiangtao Xie, Wangmeng Zuo, Lei Zhang, and Peihua Li. Deep cnns meet global covariance pooling: Better representation and generalization. *TPAMI*, 2020. 3

[24] Qilong Wang, Li Zhang, Banggu Wu, Dongwei Ren, Peihua Li, Wangmeng Zuo, and Qinghua Hu. What deep cnns benefit from global covariance pooling: an optimization perspective. In *CVPR*, 2020. 3

[25] Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. Backpropagation-friendly eigendecomposition. In *NeurIPS*, 2019. 1, 2, 3, 5, 6

[26] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 7, 8