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

This document provides additional illustrations of the SVD functions. First, we analyze the convergence property of Power Iteration method and its impact on estimating the SVD gradient (Sec. 1). Then, we explain how the Taylor polynomial gradient emerges from the Power Iteration gradient [15] and their theoretical equivalence on certain premises (Sec. 2). Finally, several superior properties of Padé approximants are introduced and proved (Sec. 3).

1. Convergence of Power Iteration

In the paper, we conjecture that Power Iteration (PI) method converges only when the first eigenvalue λ_1 is dominant. Here we analyze its convergence property and discuss the impact on the gradients associated with SVD [15]. To compute the approximate leading eigenvector $\mathbf{u}$, PI takes the iterative update:

$$\mathbf{u}^{(k)} = \frac{\mathbf{P}\mathbf{u}^{(k-1)}}{||\mathbf{P}\mathbf{u}^{(k-1)}||} \tag{1}$$

By induction on k, we have:

$$\mathbf{u}^{(k)} = \frac{\mathbf{P}^k\mathbf{u}^{(0)}}{||\mathbf{P}^k\mathbf{u}^{(0)}||},\ k \geq 1 \tag{2}$$

Since $\mathbf{P}$ is diagonalizable, the initial eigenvector can be represented by a basis function of the true eigenvectors:

$$\mathbf{u}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i \tag{3}$$

where α_i is the scalar coefficient, and $\mathbf{x}_i$ is the true eigenvector of $\mathbf{P}$. Injecting eq. (3) into eq. (2) yields:

$$\mathbf{P}^k\mathbf{u}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{P}^k\mathbf{x}_i,\ k \geq 1 \tag{4}$$

Relying on the fact $\mathbf{P}\mathbf{x}_i{=}\lambda_i\mathbf{x}_i$, eq. (4) can be re-formulated as:

$$\mathbf{P}^k\mathbf{u}^{(0)} = \sum_{i=1}^{n} \alpha_i\lambda_i^k\mathbf{x}_i = \alpha_1\lambda_1^k(\mathbf{x}_1{+}\sum_{i=2}^{n}\frac{\alpha_i}{\alpha_1}(\frac{\lambda_i}{\lambda_1})^k\mathbf{x}_i) \tag{5}$$

If the first eigenvalue λ_1 is dominant, *i.e.,* λ_i satisfies the following condition:

$$\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n \tag{6}$$

When $k{\to}\infty$, for any $\frac{\lambda_i}{\lambda_1}$, $(\frac{\lambda_i}{\lambda_1})^k$ vanishes and eq. (5) becomes $\mathbf{P}^k\mathbf{u}^{(0)}{\to}\alpha_1\lambda_1^k\mathbf{x}_1$. The constant α_1 would be cancelled by the l_2 normalization of each step. Thus, the estimation $\mathbf{u}^{(k)}$ aligns itself to the direction of the first eigenvector $\mathbf{x}_1$ and the convergence is guaranteed.

When the first eigenvalue λ_1 is not dominant (*i.e.,* $\lambda_1{=}\lambda_2$), $(\frac{\lambda_2}{\lambda_1})^k$ does not vanish and PI method cannot converge to the leading eigenvector $\mathbf{x}_1$. From eq. (5), we can also learn that the convergence rate depends on the ratio of the first two eigenvalues $\frac{\lambda_2}{\lambda_1}$. The lower the ratio is, the faster the convergence would be. When $\frac{\lambda_2}{\lambda_1}$ is close to or equal to 1 (see Fig. 3 in the paper), PI does not well approximate the leading eigenvector within limited iterations. As a consequence, the associated eigenvalue and gradients would be poorly estimated.

2. Relation between PI and Taylor Gradient

Wang *et. al.* [15] proposed to use PI method to compute the associated SVD gradients. They did not explicitly relate PI with the Taylor polynomial, but the relation between these two methods already emerged. To derive their connections, we first re-introduce the ordinary SVD gradients and PI gradient, then explain how Taylor polynomial gradient emerges from the former two methods, and end with the theoretical equivalence of PI and Taylor gradients on the premise that λ_1 is dominant.

Ordinary SVD Gradient. Consider the covariance matrix $\mathbf{P}$, the forward eigendecomposition is given by:

$$\mathbf{P} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{7}$$

where $\mathbf{U}$ and $\mathbf{V}$ are the corresponding eigenvector matrix and eigenvalue matrix, respectively. Given the loss function l, the partial derivative passed to $\mathbf{P}$ is computed as:

$$\frac{\partial l}{\partial \mathbf{P}} = \mathbf{U}((\mathbf{K}^T \circ (\mathbf{U}^T\frac{\partial l}{\partial \mathbf{U}})) + (\frac{\partial l}{\partial \mathbf{\Lambda}})_{\text{diag}})\mathbf{U}^T \tag{8}$$

where the skew-symmetric matrix $\mathbf{K}$ consists of elements K_{ij} defined by:

$$K_{ij} = \begin{cases} \frac{1}{\lambda_i - \lambda_j}, & i \neq j, \\ 0, & i = j. \end{cases} \tag{9}$$

Injecting eq. (9) into eq. (8) yields:

$$\frac{\partial l}{\partial \mathbf{P}} = \sum_{i=1}^{n} \sum_{j \neq i}^{n} \frac{1}{\lambda_i - \lambda_j} \mathbf{u}_j \mathbf{u}_j^T \frac{\partial l}{\partial \mathbf{u}_i} \mathbf{u}_i^T + \sum_{i=1}^{n} \frac{\partial l}{\partial \lambda_i} \mathbf{u}_i \mathbf{u}_i^T \tag{10}$$

where n is the total number of eigenvalues, $\mathbf{u}_i$ is the i-th row eigenvector of $\mathbf{U}$. The instability of the analytical gradient arises from $\lambda_i - \lambda_j$ in the denominator of the first term. If the two eigenvalues are very close or even equal, the resultant gradient tends to be infinite and cause overflow.

Power Iteration Gradient. As formulated in [17], the PI gradients can be computed as:

$$\begin{aligned} \frac{\partial l}{\partial \mathbf{P}} &= \sum_{k=0}^{K-1} \frac{\mathbf{I} - \mathbf{u}^{(k+1)} \mathbf{u}^{(k+1)T}}{||\mathbf{P}\mathbf{u}^{(k)}||} \frac{\partial l}{\partial \mathbf{u}^{(k+1)}} \mathbf{u}^{(k)T}, \\ \frac{\partial l}{\partial \mathbf{u}^{(k)}} &= \mathbf{P} \frac{\mathbf{I} - \mathbf{u}^{(k+1)} \mathbf{u}^{(k+1)T}}{||\mathbf{P}\mathbf{u}^{(k)}||} \frac{\partial l}{\partial \mathbf{u}^{(k+1)}}. \end{aligned} \tag{11}$$

[15] suggested that the initial eigenvector estimation starts with the accurate one calculated via SVD. Feeding it to Power Iteration defined in eq. (1) leads to:

$$\mathbf{u} = \mathbf{u}^{(0)} \approx \mathbf{u}^{(1)} \approx \mathbf{u}^{(2)} \approx \cdots \approx \mathbf{u}^{(K+1)} \tag{12}$$

This equation generally holds when the first eigenvalue λ_1 is dominant. By exploiting this equation in eq. (11), the gradient $\frac{\partial l}{\partial \mathbf{P}}$ can be re-written as:

$$\begin{aligned} \frac{\partial l}{\partial \mathbf{P}} = \Big(&\frac{(\mathbf{I} - \mathbf{u}\mathbf{u}^T)}{||\mathbf{P}\mathbf{u}||} + \frac{\mathbf{P}(\mathbf{I} - \mathbf{u}\mathbf{u}^T)}{||\mathbf{P}\mathbf{u}||^2} + \\ &\cdots + \frac{\mathbf{P}^K(\mathbf{I} - \mathbf{u}\mathbf{u}^T)}{||\mathbf{P}\mathbf{u}||^{K+1}} \Big) \frac{\partial l}{\partial \mathbf{u}} \mathbf{u}^T \end{aligned} \tag{13}$$

Relying on

$$\begin{aligned} \mathbf{P}^k &= \lambda_1^k \mathbf{u}_1 \mathbf{u}_1^T + \lambda_2^k \mathbf{u}_2 \mathbf{u}_2^T + \cdots + \lambda_3^k \mathbf{u}_3 \mathbf{u}_3^T, \\ ||\mathbf{P}\mathbf{u}||^k &= ||\lambda^k \mathbf{u}||^k = \lambda^k. \end{aligned} \tag{14}$$

The gradient in eq. (13) can be further formulated as:

$$\begin{aligned} \frac{\partial l}{\partial \mathbf{P}} &= \Big(\frac{\sum_{i=2}^{n} \mathbf{u}_i \mathbf{u}_i^T}{\lambda_1} + \cdots + \frac{\sum_{i=2}^{n} \lambda_i^K \mathbf{u}_i \mathbf{u}_i^T}{\lambda_1^{K+1}} \Big) \frac{\partial l}{\partial \mathbf{u}_1} \mathbf{u}_1^T, \\ \frac{\partial l}{\partial \mathbf{P}} &= \Big(\sum_{i=2}^{n} \Big(\frac{1}{\lambda_1} + \frac{1}{\lambda_1} (\frac{\lambda_i}{\lambda_1})^1 + \cdots + \frac{1}{\lambda_1} (\frac{\lambda_i}{\lambda_1})^K \Big) \mathbf{u}_i \mathbf{u}_i^T \Big) \frac{\partial l}{\partial \mathbf{u}_1} \mathbf{u}_1^T \\ \frac{\partial l}{\partial \mathbf{P}} &= \Big(\sum_{i=2}^{n} \frac{1}{\lambda_1} \Big(1 + (\frac{\lambda_i}{\lambda_1})^1 + \cdots + (\frac{\lambda_i}{\lambda_1})^K \Big) \mathbf{u}_i \mathbf{u}_i^T \Big) \frac{\partial l}{\partial \mathbf{u}_1} \mathbf{u}_1^T. \end{aligned} \tag{15}$$

This equation defines a geometric progression. When $k \to \infty$, we have:

$$\frac{1}{\lambda_1} \Big(1 + (\frac{\lambda_i}{\lambda_1})^1 + \cdots + (\frac{\lambda_i}{\lambda_1})^K \Big) \approx \frac{1}{\lambda_1} \frac{1}{1 - \frac{\lambda_i}{\lambda_i}} = \frac{1}{\lambda_1 - \lambda_i} \tag{16}$$

If we read the equation from right to left, it is easy to find that eq. (15) actually defines the Maclaurin series of the Taylor expansion for function $\frac{1}{\lambda_1 - \lambda_i}$. Consider the instability term $\frac{1}{\lambda_i - \lambda_j}$ of the SVD gradients in eq. (8), the Taylor polynomial gradient naturally arises if we apply the same property on the ordinary SVD gradients.

Taylor Polynomial Gradient. To obtain the Taylor polynomial gradients, we first use the property in eq. (16) to expand the skew-symmetric matrix $\mathbf{K}$ as:

$$K_{ij} = \frac{1}{\lambda_i} \cdot \frac{1}{1 - (\lambda_j / \lambda_i)} \approx \frac{1}{\lambda_i} (1 + \frac{\lambda_j}{\lambda_i} + (\frac{\lambda_j}{\lambda_i})^2 + \cdots + (\frac{\lambda_j}{\lambda_i})^K) \tag{17}$$

Here it is the Taylor expansion of function $f(x) = 1/(1-x)$ at $x = 0$ to degree K, and the higher-order term is discarded. According to Cauchy root test, the Taylor series only converges when $|\lambda_j / \lambda_i| < 1$. When $j < i$, the ratio λ_j / λ_i is larger than 1 and thus outsides the convergence radius. To avoid this issue, we can split the matrix $\mathbf{K}$ into two triangular parts and re-write the right first term of eq. (10) as:

$$\sum_{i=1}^{n} \Big(\sum_{j>i}^{n} \frac{1}{\lambda_i - \lambda_j} \mathbf{u}_j \mathbf{u}_j^T \frac{\partial l}{\partial \mathbf{u}_i} \mathbf{u}_i^T - \sum_{j<i}^{n} \frac{1}{\lambda_j - \lambda_i} \mathbf{u}_j \mathbf{u}_j^T \frac{\partial l}{\partial \mathbf{u}_i} \mathbf{u}_i^T \Big) \tag{18}$$

where the first term is the upper triangle when $j > i$ and $\lambda_j / \lambda_i \leq 1$, and the second term defines the lower triangle when $j < i$ and $\lambda_j / \lambda_i \geq 1$. As $\mathbf{K}$ is skew-symmetric , we only need to calculate the upper part, *i.e.,* the first term that can converge. Introducing the Taylor polynomial defined in eq. (17) into eq. (18), the first term is re-expressed as:

$$\sum_{j>i}^{n} \frac{1}{\lambda_i} (1 + \frac{\lambda_j}{\lambda_i} + (\frac{\lambda_j}{\lambda_i})^2 + \cdots + (\frac{\lambda_j}{\lambda_i})^K) \mathbf{u}_j \mathbf{u}_j^T \frac{\partial l}{\partial \mathbf{u}_i} \mathbf{u}_i^T \tag{19}$$

Notice that now the Taylor gradient defined in eq. (19) is quite similar with the PI gradient in eq. (15). If we set $i = 1$ for eq. (19), which represents the derivative w.r.t. the dominant eigenvector $\mathbf{v}_1$, two equations are identical and Taylor polynomial gradient is equivalent to PI gradient. This remains the same also for the other eigenvectors. The equivalence holds when the first eigenvalue λ_1 is dominant and therefore PI gradient is valid. That being said, the Taylor polynomial gradient which emerges from the ordinary SVD gradient and PI gradient is actually a more general expression of the gradient calculated by PI method.

3. Properties of Padé Approximants

We introduce three relevant properties of Padé approximants and give proofs on the first and the last theorems.

Uniqueness of Solution. Padé approximants are defined by matching a given Taylor series and have the unique solution for each matched pair. [2] gives the following theorem:

Any existing $[M/N]$ Padé approximants to their formal power series $A(x)$ has the unique solution.

This theorem is easy to be proved. Suppose there are two such Padé approximants $P(x)/Q(x)$ and $U(x)/V(x)$ of the same degree, they must satisfy $P(x)/Q(x)-U(x)/V(x)=O(x^{M+N+1})$ as both approximate the same series. Multiplying $Q(x)V(x)$ on both sides lead to $P(x)/Q(x)=U(x)/V(x)$. Since by definition $Q(0)=V(0)=1$ and both P and Q, U and V are relatively prime, we can conclude that the supposedly different approximants are the same.

Special Case of Continued Fraction. Continued fraction is known as one of the best rational approximation techniques [13]. A general continued fraction expression takes the form as:

$$C = b_0 + \sum_{i=1}^{\infty} \frac{a_i}{b_i+} = b_0 + \frac{a_1}{b_1 + \frac{a_2}{b_2+\frac{a_3}{b_3+\dots}}} \tag{20}$$

where a_i and b_i are partial numerator and denominator, respectively. If the n^{th} convergent of eq. (20) is denoted as a fraction A_n/B_n. , the recursive relations of the successive convergents can be expressed as

$$\begin{aligned} A_{n+1} &= b_{n+1}A_n + a_{n+1}A_{n-1}, \\ B_{n+1} &= b_{n+1}B_n + a_{n+1}B_{n-1}. \end{aligned} \tag{21}$$

Note that eq. (21) resembles the recursive relation of numerator $P_M(x)$ and denominator $Q_N(x)$ of Padé approximants. Therefore, Padé approximants can be viewed as a special case of continued fraction. Particularly in diagonal case, we have: The successive convergents of Jacobi-type continued fractions can be given by corresponding diagonal sequence $[1/0], [2/1], \dots$ of Padé approximants. This theorem has been extensively proved in the literature [2, 1, 4]. It allows us to associate Padé approximants to continued fraction and calculate Padé coefficients recursively using eq. (21). This kind of recursive computation is usually more stable than solving linear equations, as the solution of linear equations might be close to the singularities of Toeplitz matrices [6].

Enlarged Convergence Range. We present the following theorem in the paper without proof: If the function $f(z)$ is a Stieltjes transform $f(z)=\int_a^b \frac{1}{z-x} du(x)$ of a compactly supported measure $u(x)$ 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 theorem describes the orthogonality constraints and convergence property of diagonal Padé approximants. The orthogonality where $\int_a^b Q_N(x)P_{N+1}d\mu(x)=0$ is not easy to be proved, the readers are kindly suggested to refer to [14] for a detailed review. Taking the orthogonality of P_{N+1} as the condition in hand, we give a concise proof on the convergence.

Proof. Given that 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 $[a,b]$, the denominator polynomial in the Padé approximation is orthogonal polynomial for the measure μ on the interval $[a,b]$:

$$\int_a^b x^k Q_N(x) d\mu(x) = 0 \tag{22}$$

The numerator polynomial is given by

$$P_{N+1}(z) = \int_a^b \frac{Q_N(z) - Q_N(x)}{z-x} d\mu(x) \tag{23}$$

By normalizing eq. (22), the approximation error of Padé approximants is calculated as:

$$f(z) - \frac{P_{N+1}(z)}{Q_N(z)} = \frac{1}{Q_N(z)} \int_a^b \frac{Q_N(x)}{z-x} d\mu(x) \tag{24}$$

Observe that

$$\begin{aligned} &Q_N(z) \int_a^b \frac{Q_N(x)}{z-x} d\mu(x) = \\ &\int_a^b \frac{Q_N(x)(Q_N(z) - Q_N(x))}{z-x} d\mu(x) + \int_a^b \frac{Q_N^2(x)}{z-x} d\mu(x) \end{aligned} \tag{25}$$

By orthogonality constraint the first integral on the right side vanishes. The error then becomes:

$$f(z) - \frac{P_{N+1}(z)}{Q_N(z)} = \frac{1}{Q_N^2(z)} \int_a^b \frac{Q_N^2(x)}{z-x} d\mu(x) \tag{26}$$

Now the error contains two parts $\frac{1}{Q_N^2(z)}$ and $\int_a^b \frac{Q_N^2(x)}{z-x} d\mu(x)$. The integral term is actually a Markov function for the probability measure $Q_N^2(x)d\mu(x)$ when Q_N is orthonormal. It can be estimated by a strictly positive distance measure defined as:

$$d_K := inf\{|z-x| : z \in K, x \in [a,b]\} \tag{27}$$

where K is a compact set that z belongs to. Then we have:

$$\left| \int_a^b \frac{Q_N^2(x)}{z-x} d\mu(x) \right| \le \int_a^b \frac{Q_N^2(x)}{|z-x|} d\mu(x) \le \frac{1}{d_K} \tag{28}$$

This bound is independent of the polynomial degree N. So the convergence is completely determined by the asymptotic behavior of Q_N. By measuring the logarithm energy for the leading coefficients of Q_N, [14] further shows that

$$\lim_{N\to\infty} |Q_N(z)|^{\frac{1}{N}} = \frac{4}{b-a} \exp\Big(- \int_a^b \log \frac{1}{|z-x|} d\mu_e(x) \Big) \tag{29}$$

where μ_e is a unique probability measure on $[a,b]$, and the right hand side is larger than 1 when z moves away from $[a,b]$. Let r denotes the right hand side of eq. (29), the error in eq. (26) can be re-formulated as:

$$\lim_{N\to\infty} \left| f(z) - \frac{P_{N+1}(z)}{Q_N(z)} \right| = \frac{1}{r^2},\ r > 1 \tag{30}$$

We can conclude that the diagonal Padé approximants have exponential convergence in $[a,b]$. □

This document introduces the experimental settings, some analyses on the SVD meta-layers, and extra ablation studies. First, we present the experimental settings in Sec. 4. Subsequently, Sec. 5 and Sec. 6 justify the degree selection of the Taylor series and compare the upper bound of the gradient for each SVD method, respectively. Finally, Sec. 7 describes the results of ablation studies on the random seeds and warm-up epochs.

4. Experimental Settings

Models and Datasets. Following [11, 10], we first take AlexNet [9] and ResNet-50 [5] as the backbones and conduct experiments on ImageNet 2012 [3] for the large-scale visual recognition. This dataset has 1.28M images for training and 50K images for testing. The covariance pooling meta-layer is inserted before the fully-connected layer of each model. For AlexNet architecture, the outputs of convolutional layers are $13\times13\times256$ tensor. For ResNet architecture, we add 1×1 convolution to squeeze the channels of global representation from 2048 to 256. Therefore, the covariance matrices of both networks are of the same size 256×256. Since the covariance is a symmetric matrix, only the upper triangular part is taken and passed to the fully-connected layer. After training GCP models on ImageNet, we then conduct experiments on the task of Fine-Grained Visual Categorization (FGVC). The pre-trained ResNet-50 with different GCP meta-layers using the hybrid training strategy are fine-tuned on three popular fine-grained benchmarks, *i.e.,* Caltech Birds (Birds) [16], Stanford Dogs (Dogs) [7], and Stanford Cars (Cars) [8]. The Birds dataset contains $11,788$ images belonging to 200 species. The Dogs dataset includes $20,580$ images of 120 breeds of dogs, and the Cars dataset consists of $16,185$ images from 196 classes of cars.

Implementation Details. All the source codes are implemented in Pytorch. Except that the forward eigendecomposition is performed on CPU for faster speed, the other operations are conducted on GPU. For the SVD-TopN method, we keep the top 200 out of 256 eigenvalues. The maximum gradient for the SVD-Trunc method is limited to 10^{10}. We set the iteration times as 10 for the SVD-Newton method. For SVD-Taylor and SVD-Padé, we truncate the Taylor series to degree 100 and match the diagonal Padé approximants also to degree 100, respectively. During the forward pass, the eigenvalues that are smaller than EPS, *i.e.,* the smallest positive number that the data precision can represent, are set as EPS for numerical stability.

ImageNet Setting. On ImageNet, the AlexNet is trained for 30 epochs with an initial learning rate set as $10^{-1.1}$. The learning rate decays by 10 every 10 epochs. For training ResNet, we use the same learning rate to train for 60 epochs but decays by 10 at epoch 30 and epoch 45. When applying the hybrid training strategy on AlexNet, all the methods are warmed up for 1 epoch when switching to SVD methods. The warm-up epoch for ResNet is set as 2. The batch size is set to 128 for AlexNet and 256 for ResNet. We use SGD for optimization, with momentum of 0.9 and weight decay of 0.0001 for ResNet and 0.0005 for AlexNet. The network parameters are randomly initialized for both AlexNet and ResNet. During training, the images are resized to 256×256 and then cropped to 224×224, with random horizontal flip augmentation. The inference is conducted on the 224×224 centered crop from the test image.

FGVC Setting. For FGVC datasets, the images are first resized to 448×448 and then fed into the network. The 1000-d fully-connected layer of the original model is changed to fit the number of classes. The model is trained using SGD with momentum 0.9 and weight decay 0.0001. The batch size is set 10, and the training lasts 50 epochs for all the datasets. The learning rate is set as 6×10^{-3} for the fully-connected layer and 1.2×10^{-3} for the other layers. We make the inference on the 448×448 centered crop of the test image.

5. Choosing Degree of Taylor Series

Both SVD-Taylor and SVD-Padé need to match the truncated Taylor series of degree K. For SVD-Taylor, K determines the upper bound of gradient approximation $(K+1)/\lambda_i$ and the discarded higher-order term $\sum_{i=K+1}^{\infty}(\lambda_j/\lambda_i)^i$. As can be observed from Table 1, a small K will yield poor approximation, but a large K increases the computation time and still fails to well approximate values close to the convergence boundary ($\lambda_j/\lambda_i \approx 1$). Unless K is set very large (*e.g.,* 10^{20}), gradients near the polar singularities can not closely estimated. Thus, choosing an appropriate K can substantially influence the gradient approximation. For SVD-Padé, K has a slight effect on the approximation due to the enlarged convergence range of Padé approximants (see Table 2). The degree K is set as 100 through cross-validation for the best performances of SVD-Taylor. We also choose K=100 for SVD-Padé to make sure that both methods agree up to the same degree.

Table 1. Approximation error of Taylor polynomial of different degrees in double precision.

Degree \ λ_j/λ_i	0.1	0.3	0.5	0.7	0.9	0.99	0.999
50	9e-19	7e-18	9e-16	4e-8	5e-2	60	950
100	9e-19	7e-18	2e-21	8e-16	2e-4	36	904
200	9e-19	7e-18	2e-21	4e-17	6e-9	13	817
300	9e-19	7e-18	1e-21	4e-17	1e-13	5	740

6. Upper Bound of Gradient

We first describe how the upper bound is attained for each method in detail and then discuss their behaviors.

Table 2. Approximation error of diagonal Padé approximants of different degrees in double precision.

Degree \ λ_j/λ_i	0.1	0.3	0.5	0.7	0.9	0.99	0.999
50	2e-18	3e-17	2e-20	6e-17	3e-16	1e-13	1e-12
100	9e-19	5e-18	1e-21	5e-17	3e-16	8e-13	3e-10
200	2e-18	1e-17	6e-21	6e-18	1e-15	1e-13	2e-10
300	1e-18	8e-18	6e-21	5e-17	2e-15	1e-13	5e-10

SVD-Padé. Both SVD-Padé and SVD-Taylor decompose the gradient function as:

$$K_{ij} = \frac{1}{\lambda_i - \lambda_j} = \frac{1}{\lambda_i} \frac{1}{1 - \lambda_j/\lambda_i} \tag{31}$$

The term $\frac{1}{1-\lambda_j/\lambda_i}$ can be viewed as the function $f(x)=\frac{1}{1-x}$ and we use Padé approximants to approximate it. Since the function $f(x)$ is monotonically increasing in the range $[0,1]$, the upper bound of Padé approximants is reached at $x=1$, *i.e.,* when the two eigenvalues λ_i and λ_j are identical. The upper bound for eq. (31) can be represented as:

$$|K_{ij}| \leq \left|\frac{1}{\lambda_i}\right| \cdot \left| \frac{\sum_{m=0}^{M} p_m}{1 + \sum_{n=1}^{N} q_n} \right| \tag{32}$$

The second fraction on the right side denotes the maximal value of Padé approximants when $\frac{\lambda_j}{\lambda_i}=1$. We compute this result as $6.48e20$. Now the upper bound of gradient depends on $\frac{1}{\lambda_i}$. Suppose λ_i and λ_j simultaneously equal to EPS, the resultant upper bound of SVD-Padé is attained. The bound is calculated as $6.00e36$ and it happens only when the two eigenvalues simultaneously have the minimum possible value ($\lambda_i=\lambda_j=$EPS).

SVD-Taylor. For SVD-Taylor, the upper bound relies on the truncated degree of Taylor series K and the minimum value of λ_i. Specifically, the Taylor polynomial is a bounded estimation:

$$\frac{1}{1 - \lambda_j/\lambda_i} \approx 1 + \frac{\lambda_j}{\lambda_i} + (\frac{\lambda_j}{\lambda_i})^2 + \cdots + (\frac{\lambda_j}{\lambda_i})^K \leq K+1 \tag{33}$$

The equality is taken if $\lambda_i=\lambda_j$. Combining eq. (33) with eq. (31), the analytical form of the upper bound can be derived as:

$$|K_{ij}| \leq \left|\frac{K+1}{\lambda_i}\right| \tag{34}$$

Similar with SVD-Padé, when $\lambda_i=\lambda_j=$EPS, the upper bound is attained as $4.55e17$.

SVD-Trunc. As we directly truncate the gradient K_{ij} by a large constant T for SVD-Trunc, the upper bound of gradient is equal to T. The truncation and also the upper bound are triggered when $\left|\frac{1}{\lambda_i-\lambda_j}\right| \geq$T.

SVD-TopN. For SVD-TopN, the upper bound of gradient is very likely to happen between the last kept eigenvalue λ_N and the first abandoned eigenvalue λ_{N+1}. As λ_{N+1} is truncated to zero, the bound takes the form $\frac{1}{\lambda_N}$. The maximal value is reached when $\lambda_N=$EPS.

SVD-Newton. As the iterative matrix-matrix product is involved in the backward algorithm of Newton-Schulz iteration, the upper bound for the SVD-Newton method can not be derived. But from the empirical observation on the effective β-smoothness [12] (see Fig. 4 right in the paper), the gradient is very smooth and the upper bound is expected to be similar with SVD-Trunc.

SVD. The ordinary SVD gradient takes the form $\frac{1}{\lambda_i-\lambda_j}$. When the two eigenvalues are equal, the gradient will explode and go to infinity.

Table. 3 summarizes the upper bound of gradient K_{ij} for each SVD variant and their happening conditions. Compared with the ordinary SVD gradients, these SVD remedies reduce both the magnitude and the occurrence of the upper bound. Our proposed SVD-Padé allows for the largest gradient upper bound, but the maximal value is still acceptable in the double precision ($<1.79e308$). Even for the single precision ($<3.40e38$), the gradient is also numerically stable and allowed. This can ensure that the SVD-Padé meta-layer is compatible with the backbone either in single or double precision. The compatibility also shows the possibility that our SVD meta-layers can be trained by the recent advanced mixed-precision training techniques (*e.g.,* Pytorch 1.8 and Nvidia Apex 1.0) for acceleration and stability.

Table 3. 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	/	$\lambda_i = \lambda_j$

7. Ablation Studies

Impact of Random Seed. We measure the impact of random seeds by having 5 runs for SVD-Padé meta-layer on AlexNet using the standalone training strategy. Different random seeds do influence the network in the early epochs, but the impact gets weakened in the later stage. The final error fluctuates within 0.1%. This variation would not shake our deductions, as our SVD-Padé meta-layer outperforms the other SVD remedies by at least 0.2%. We expect that the fluctuation would be similar or smaller for ResNet and hybrid training strategy.

Impact of Warm-up Epochs. We take our proposed SVD-Padé as the meta-layer and evaluate the impact of warm-up epochs when using the hybrid training strategy. As can be seen from Table 4, two epochs achieve the best results in the final error. If warming up for more epochs, no obvious performance gain is observed in the final error but the best

error continues to improve. It is also worth mentioning that the performance is still competitive even without any warm-up training. We set to 1 epoch for AlexNet in order not to introduce heavy burdens on the training process.

Table 4. Validation error of AlexNet using SVD-Padé meta-layer and hybrid the training strategy with various warm-up epochs. The best three results are highlighted in red, blue, and green.

Settings	Final Error (%)		Best error (%)	
	top-1	top-5	top-1	top-5
no warm-up	47.89	23.82	47.75	23.63
1 epoch	47.76	23.48	47.63	23.21
2 epochs	47.70	23.39	47.59	23.23
3 epochs	47.95	23.57	47.54	23.14
iSQRT-COV [10]	47.95	23.64	47.81	23.54

References

[1] George Allen Baker and John L Gammel. *The Padé approximant in theoretical physics*. Academic Press, 1970. 3

[2] George A Baker Jr. The theory and application of the padé approximant method. Technical report, Los Alamos Scientific Lab., Univ. of California, N. Mex., 1964. 3

[3] 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. 5

[4] A George Jr et al. *Essentials of Padé approximants*. Elsevier, 1975. 3

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

[6] JOSEF Kallrath. On rational function techniques and pade approximants. an overview, 2002. 3

[7] 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. 5

[8] 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. 5

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

[10] 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. 5, 7

[11] Peihua Li, Jiangtao Xie, Qilong Wang, and Wangmeng Zuo. Is second-order information helpful for large-scale visual recognition? In *ICCV*, 2017. 5

[12] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003. 6

[13] Michael James David Powell et al. *Approximation theory and methods*. Cambridge university press, 1981. 3

[14] Walter Van Assche. Padé and hermite-padé approximation and orthogonality. *arXiv preprint math/0609094*, 2006. 3

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

[16] 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. 5

[17] Mang Ye, Andy J Ma, Liang Zheng, Jiawei Li, and Pong C Yuen. Dynamic label graph matching for unsupervised video re-identification. In *ICCV*, 2017. 2