Supplementary Material for ReDro: Efficiently Learning Large-sized SPD Visual Representation

Saimunur Rahman[1,2][0000−0002−5250−5612], Lei Wang[1][0000−0002−0961−0441], Changming Sun[2][0000−0001−5943−1989], and Luping Zhou[3][0000−0003−1065−6604]

[1] VILA, School of Computing and Information Technology, University of Wollongong, NSW 2522, Australia
[2] CSIRO Data61, PO Box 76, Epping, NSW 1710, Australia
[3] School of Electrical and Information Engineering, University of Sydney, NSW 2006, Australia
sr801@uowmail.edu.au; leiw@uow.edu.au; changming.sun@csiro.au; luping.zhou@sydney.edu.au

1 Proof of equation (7) in the main text

In this section, we provide the proof of equation (7) mentioned in Section 3.2 (second paragraph) of the main text. Let us begin the proof by recalling equation (6) in the main text,

$$\mathbf{X} \to \underbrace{\mathbf{P}\mathbf{X}}_{\mathbf{Y}} \to \underbrace{(\mathbf{Y}\mathbf{Y}^\top) \circ \mathbf{S}}_{\mathbf{C}_b} \to \underbrace{(\mathbf{P}^\top \mathbf{C}_b \mathbf{P})}_{\mathbf{A}(\text{Auxiliary})} \to \underbrace{f(\mathbf{A})}_{\mathbf{Z}} \to \cdots \text{layers} \cdots \to \underbrace{J(\mathbf{X})}_{\text{Objective}} \quad (1)$$

According to equation (1), J is a composite function that has been applied to $\mathbf{X}$ and it can be equally expressed as a function of each of the intermediate variables as follows.

$$J(\mathbf{X}) = J_1(\mathbf{Y}) = J_2(\mathbf{C}_b) = J_3(\mathbf{A}) = J_4(\mathbf{Z}) \quad (2)$$

By the rules of differentiation, the following results can be obtained

$$\delta\mathbf{Y} = \mathbf{P}\delta\mathbf{X}, \quad (3)$$

$$\delta\mathbf{C}_b = [(\delta\mathbf{Y})\mathbf{Y}^\top + \mathbf{Y}(\delta\mathbf{Y})^\top] \circ \mathbf{S}, \quad (4)$$

$$\delta\mathbf{A} = \mathbf{P}^\top \delta\mathbf{C}_b \mathbf{P} \quad (5)$$

By the differentiation rule of a scalar-valued matrix function, we know that

$$\delta J = \left\langle \text{vec}\left(\frac{\partial J_3}{\partial \mathbf{A}}\right), \text{vec}(\delta\mathbf{A}) \right\rangle = \text{trace}\left(\left(\frac{\partial J_3}{\partial \mathbf{A}}\right)^\top \delta\mathbf{A}\right) \quad (6)$$

where $\text{vec}(\cdot)$ denotes the vectorization of a matrix and $\langle \cdot, \cdot \rangle$ denotes the inner product. Combing the result with $\delta\mathbf{A} = \mathbf{P}^\top \delta\mathbf{C}_b\mathbf{P}$ in equation (5), we can obtain

$$\begin{aligned}\delta J = \text{trace}\left(\left(\frac{\partial J_3}{\partial \mathbf{A}}\right)^\top \delta\mathbf{A}\right) &= \text{trace}\left(\left(\frac{\partial J_3}{\partial \mathbf{A}}\right)^\top \mathbf{P}^\top \delta\mathbf{C}_b \mathbf{P}\right) \\ &= \text{trace}\left(\left(\mathbf{P}\frac{\partial J_3}{\partial \mathbf{A}}\mathbf{P}^\top\right)^\top \delta\mathbf{C}_b\right) \\ &= \text{trace}\left(\left(\frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top \delta\mathbf{C}_b\right)\end{aligned} \tag{7}$$

The last equality holds because from equations (2) and (6) we know that δJ can also be written as $\text{trace}\big(\big(\frac{\partial J_2}{\partial \mathbf{C}_b}\big)^\top \delta\mathbf{C}_b\big)$. Noting that Equation (7) is true for any $\delta\mathbf{C}_b$, we can therefore derive that

$$\frac{\partial J_2}{\partial \mathbf{C}_b} = \mathbf{P}\frac{\partial J_3}{\partial \mathbf{A}}\mathbf{P}^\top \tag{8}$$

Note that $\frac{\partial J_3}{\partial \mathbf{A}}$ can be computed as $\frac{\partial J_3}{\partial \mathbf{A}} = \hat{\mathbf{U}}_b\big(\mathbf{G} \circ (\hat{\mathbf{U}}_b^\top \frac{\partial J_4}{\partial \mathbf{Z}} \hat{\mathbf{U}}_b)\big)\hat{\mathbf{U}}_b^\top$, where the (i,j)th entry g_{ij} of matrix $\mathbf{G}$ is defined as $\frac{f(\lambda_i)-f(\lambda_j)}{\lambda_i-\lambda_j}$ if $\lambda_i \neq \lambda_j$ and $f'(\lambda_i)$ otherwise, where λ_i is the ith diagonal element of $\hat{\mathbf{D}}_b$. Readers are referred to [1] and [2] for the proof of $\frac{\partial J_3}{\partial \mathbf{A}}$.

Again, combing $\delta J = \text{trace}\big(\big(\frac{\partial J_2}{\partial \mathbf{C}_b}\big)^\top \delta\mathbf{C}_b\big)$ with $\delta\mathbf{C}_b = [(\delta\mathbf{Y})\mathbf{Y}^\top + \mathbf{Y}(\delta\mathbf{Y})^\top] \circ \mathbf{S}$ in equation (4), it can be obtained that

$$\text{trace}\left(\left(\frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top \delta\mathbf{C}_b\right) = \text{trace}\left(\left(\frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top ([(\delta\mathbf{Y})\mathbf{Y}^\top + \mathbf{Y}(\delta\mathbf{Y})^\top] \circ \mathbf{S})\right) \tag{9}$$

By the identity that $\text{trace}(\mathbf{A}^\top(\mathbf{B} \circ \mathbf{C})) = \text{trace}((\mathbf{B} \circ \mathbf{A})^\top \mathbf{C})$, we can obtain

$$\text{trace}\left(\left(\frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top \delta\mathbf{C}_b\right) = \text{trace}\left(\left(\mathbf{S} \circ \frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top ((\delta\mathbf{Y})\mathbf{Y}^\top + \mathbf{Y}(\delta\mathbf{Y})^\top)\right) \tag{10}$$

Denoting $\big(\mathbf{S} \circ \frac{\partial J_2}{\partial \mathbf{C}_b}\big)$ with $\mathbf{Q}$, and applying the identity that $\text{trace}(\mathbf{A}+\mathbf{B}) = \text{trace}(\mathbf{A})+\text{trace}(\mathbf{B})$, $\text{trace}(\mathbf{ABC}) = \text{trace}(\mathbf{CAB})$ and $\text{trace}(\mathbf{ABC}) = \text{trace}((\mathbf{ABC})^\top)$, we can further simplify equation (10) as

$$\begin{aligned}\text{trace}\left(\left(\frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top \delta\mathbf{C}_b\right) &= \text{trace}\left(\mathbf{Q}^\top((\delta\mathbf{Y})\mathbf{Y}^\top + \mathbf{Y}(\delta\mathbf{Y})^\top)\right) \\ &= \text{trace}\left(\left((\mathbf{Q}+\mathbf{Q}^\top)\mathbf{Y}\right)^\top \delta\mathbf{Y}\right) \\ &= \text{trace}\left(\left(\frac{\partial J_1}{\partial \mathbf{Y}}\right)^\top \delta\mathbf{Y}\right)\end{aligned} \tag{11}$$

Again, because we know δJ can also be expressed as $\text{trace}\big(\big(\frac{\partial J_1}{\partial \mathbf{Y}}\big)^\top \delta\mathbf{Y}\big)$ and the last result is valid for any $\delta\mathbf{Y}$, it can be obtained that

$$\frac{\partial J_1}{\partial \mathbf{Y}} = (\mathbf{Q}+\mathbf{Q}^\top)\mathbf{Y} \tag{12}$$

By substituting the value of $\mathbf{Q}$ in equation (12) and using the identitiy $(\mathbf{A} \circ \mathbf{B})^\top = \mathbf{A}^\top \circ \mathbf{B}^\top$, we then obtain

$$\begin{aligned}\frac{\partial J_1}{\partial \mathbf{Y}} &= \left(\left(\mathbf{S} \circ \frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top + \left(\left(\mathbf{S} \circ \frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top\right)^\top\right)\mathbf{Y} \\ &= \left(\mathbf{S} \circ \left(\frac{\partial J_2}{\partial \mathbf{C}_b} + \left(\frac{\partial J_2}{\partial \mathbf{C}_b}\right)^\top\right)\right)\mathbf{Y}\end{aligned} \tag{13}$$

Again, combing $\delta J = \text{trace}\left(\left(\frac{\partial J_1}{\partial \mathbf{Y}}\right)^\top \delta\mathbf{Y}\right)$ with $\delta\mathbf{Y} = \mathbf{P}\delta\mathbf{X}$ in equation (3), it can be obtained that

$$\begin{aligned}\text{trace}\left(\left(\frac{\partial J_1}{\partial \mathbf{Y}}\right)^\top \delta\mathbf{Y}\right) &= \text{trace}\left(\left(\frac{\partial J_1}{\partial \mathbf{Y}}\right)^\top \mathbf{P}\delta\mathbf{X}\right) \\ &= \text{trace}\left(\left(\mathbf{P}^\top \frac{\partial J_1}{\partial \mathbf{Y}}\right)^\top \delta\mathbf{X}\right) \\ &= \text{trace}\left(\left(\frac{\partial J}{\partial \mathbf{X}}\right)^\top \delta\mathbf{X}\right)\end{aligned} \tag{14}$$

Again, because we know δJ can also be expressed as $\text{trace}\left(\left(\frac{\partial J}{\partial \mathbf{X}}\right)^\top \delta\mathbf{X}\right)$ and the last result is valid for any $\delta\mathbf{X}$, it can be obtained that

$$\frac{\partial J}{\partial \mathbf{X}} = \mathbf{P}^\top \frac{\partial J_1}{\partial \mathbf{Y}} \tag{15}$$

This completes the proof. In equation (7) in the main text, we omit the subscript of J in its gradients with respect to $\mathbf{A}, \mathbf{C}_b$ and $\mathbf{Y}$.

2 Dataset information

In this section, we provide the dataset information mentioned in Section 4 (first paragraph) of the main text. We perform experiments on four widely used public image datasets, namely, MIT Indoor [8], Stanford Cars [3], Caltech-UCSD Birds (CUB 200-2011) [10] and FGVC-Aircraft [7] to demonosrate the performance of ReDro. Figure 1 shows the sample images from these datasets and more details are given below:

MIT Indoor dataset is one of the most widely used datasets in the literature for scene classification. It has a total of 15,620 images and 67 classes. Each image class contains a minimum of 100 images. The images are collected from various types of stores (e.g., grocery, bakery), private places (e.g., bedroom and living room), public places (e.g., prison cell, bus, library), recreational places (e.g. restaurant, bar, cinema hall) and working environments (e.g. office, studio).

Caltech-UCSD Birds or simply 'Birds' is one of the most reported datasets in fine-grained image classification (FGIC) literature. It has a total of 11,788 images

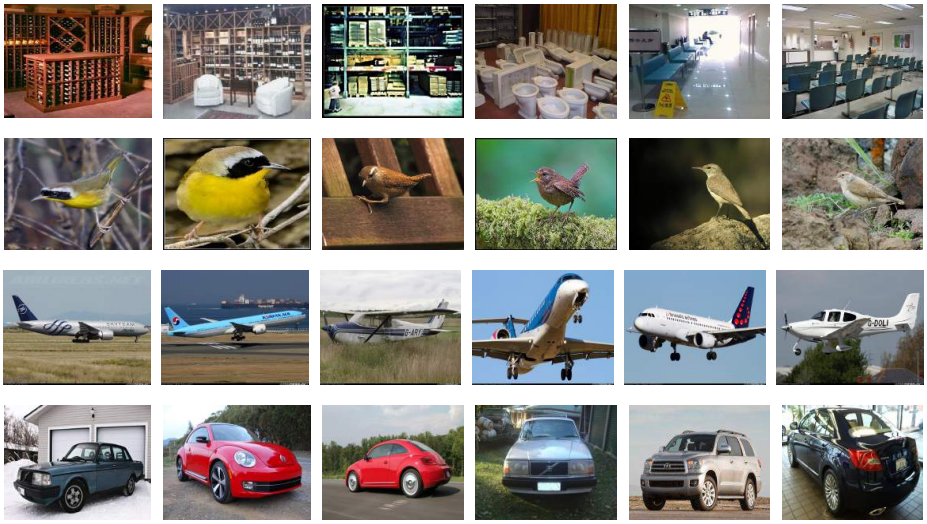

Fig. 1. Sample images from the datasets used in our experiments. Rows 1, 2, 3 and 4 have the images from MIT Indoor, Caltech-UCSD Birds, FGVC-Aircraft and Stanford Cars datasets, respectively.

and 200 image classes. There are subtle differences between these classes and they are indistinguishable by human observers. This dataset comes with bounding box annotations; however, we do not use any annotations in our experiments.

FGVC-Aircraft or 'Aircraft' dataset is relatively a smaller dataset but widely used in recent FGIC methods. It has only 10,000 images distributed among 100 aircraft classes, and each class has precisely 100 images. Similar to Birds, the classes have subtle differences between them and are hard for humans to distinguish from each other. However, compared to Birds, the size of airplanes, i.e., objects, are relatively larger in each image.

Stanford Cars or simply 'Cars' has a total of 16,185 images and 196 classes. The classes are organized as per the car production year, car manufacturer and car model. Cars dataset has relatively smaller objects, i.e., cars, than those of the airplane dataset. Furthermore, the objects are appeared in cluttered backgrounds. Table 1 gives a more concise summary of the datasets.

3 Implementation of ReDro

In this section, we provide the implementation steps of ReDro. Following the Algorithm 1 in the main text, we implement ReDro in four phases:

1. *Channel permutation.* Given feature channels from the last convolutional layer of the backbone CNN model and a permutation matrix generated at a run-time, we perform permutation of the feature channels with the permutation matrix.

Table 1. Summary of datasets

Dataset	Total classes	Total images	Predefined protocol		Major difficulty
			Training images	Testing images	
MIT Indoor	67	6700	5,360	1,340	difficult environment
Birds	200	11,788	5,994	5,794	subtle class difference
Aircraft	100	10,000	6,600	3,400	subtle class difference
Cars	196	16,185	8,144	8,041	cluttered background

2. *Block-diagonal matrix computation.* The permuted feature channels in phase 1 are partitioned into k equally sized groups and on each group, a small covariance matrix is computed. Next, eigen-decomposition is computed for each of the small covariance matrices, and the resultant eigenvalues and eigenvectors are assembled in a block-diagonal manner.
3. *Back-permutation of eigenvectors.* Using the permutation matrix from phase 1, we then permute back the eigenvectors obtained in phase 2 to make the random permutation in ReDro transparent to the subsequent network layers.
4. *Matrix normalisation.* Given the eigenvalues and eigenvectors from phases 2 and 3, we then apply the matrix normalisation.

Depending on the SPD visual representation methods (used in Section 4.3 in the main text) and the availability of source code, we implement the above four

Table 2. Comparison of computational time (in second) for covariance estimation and matrix normalisation by using or not using the proposed ReDro scheme. The forward propagation time (F.P.), backward propagation time (B.P.) and the sum of forward and backward propagation time (Total) are reported. The four methods to the left represent the case not using ReDro. The case using ReDro is implemented upon DeepCOV [1] with various k. The boldface shows that ReDro saves computational time

Matrix Dimension	Mode	No ReDro used				DeepCOV [1] using ReDro			
		MPN -COV [5]	Deep -COV [1]	IBCNN [6]	iSQRT -COV [4]	with $k=2$	with $k=4$	with $k=8$	with $k=16$
128×128	Total	0.004	0.004	0.006	0.001	0.007	0.009	0.011	0.015
	F.P.	0.003	0.003	0.003	0.0006	0.004	0.006	0.008	0.012
	B.P.	0.001	0.001	0.003	0.0006	0.003	0.003	0.003	0.003
256×256	Total	0.013	0.013	0.014	0.006	0.011	0.011	0.013	0.020
	F.P.	0.011	0.011	0.011	0.0053	0.007	0.007	0.009	0.016
	B.P.	0.002	0.002	0.003	0.0012	0.004	0.004	0.004	0.004
512×512	Total	0.031	0.031	0.032	0.030	0.030	**0.022**	**0.023**	0.030
	F.P.	0.025	0.025	0.025	0.0270	0.023	**0.015**	**0.016**	0.023
	B.P.	0.006	0.006	0.007	0.0033	0.007	0.007	0.007	0.007
1024×1024	Total	0.097	0.097	0.121	0.097	**0.090**	**0.076**	**0.056**	**0.062**
	F.P.	0.089	0.089	0.111	0.0872	**0.072**	**0.058**	**0.039**	**0.045**
	B.P.	0.008	0.008	0.010	0.0095	0.018	0.018	0.017	0.017

phases by MatConvNet [9] using one or multiple layers. *Our source code containing the layer implementations, experimental frameworks and dataset protocols will be available online.*

4 Computational advantage of ReDro

In this section, we provide the forward and backward propagation time of the methods compared in Table 1 of the main text. Table 1 in this supplement material shows the forward propagation time (indicated with "F.P.") and backward propagation time (indicated with "B.P.") alongside with the total (i.e., forward propagation+backward propagation) time (indicated with "Total").

From the table, in addition to the discussions provided in Section 4.1 in the main text, we can observe that ReDro saves the computation time of forward propagation in learning large-sized covariance matrices, i.e., of the size of 512×512 and 1024×1024. Meanwhile, note that as expected, ReDro will not save any computation time for backward propagation since it is only applied to deal with the eigen-decomposition in the forward propagation.

References

1. Engin, M., Wang, L., Zhou, L., Liu, X.: DeepKSPD: Learning kernel-matrix-based spd representation for fine-grained image recognition. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 612–627. Springer (2018)
2. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. In: Proceedings of the International Conference on Computer Vision. pp. 2965–2973. IEEE (2015)
3. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D object representations for fine-grained categorization. In: Proceedings of the International Conference on Computer Vision Workshops. pp. 554–561. IEEE (2013)
4. Li, P., Xie, J., Wang, Q., Gao, Z.: Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 947–955. IEEE (2018)
5. Li, P., Xie, J., Wang, Q., Zuo, W.: Is second-order information helpful for large-scale visual recognition? In: Proceedings of the International Conference on Computer Vision. pp. 2070–2078. IEEE (2017)
6. Lin, T.Y., Maji, S.: Improved Bilinear Pooling with CNNs. arXiv preprint arXiv:1707.06772 (2017)
7. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013)
8. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 413–420. IEEE (2009)
9. Vedaldi, A., Lenc, K.: MatConvNet: Convolutional Neural Networks for Matlab. In: Proceedings of the 23rd ACM International Conference on Multimedia. pp. 689–692 (2015)
10. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-UCSD Birds 200. Tech. Rep. CNS-TR-2010-001, California Institute of Technology (2010)