Get the Best of Both Worlds: Improving Accuracy and Transferability by Grassmann Class Representation Supplementary Material

Haoqi Wang* Zhizhong Li* Wayne Zhang†
haoqi.wang@epfl.ch {lizz,wayne.zhang}@sensetime.com

In this supplementary material, we provide experimental results and implementation details that were omitted in the main paper due to space limitations.

A. An Alternative Form of Riemannian SGD

As discussed in Section 4.2, an important ingredient of the geometric optimization algorithm is to move a point in the direction of a tangent vector while staying on the manifold (*e.g.*, see the example in Fig. 1). This is accomplished by the *retraction* operation (please refer to [1, Section 4.1] for its mathematical definition). In the fourth step of Alg. 1, the retraction is implemented by computing the geodesic curve on the manifold that is tangent to the vector $\boldsymbol{M}^{(t)}$. An alternative implementation of retraction other than moving parameters along the geodesic is to replace step 4 with the Euclidean gradient update $\boldsymbol{S}^{(t+1)} \leftarrow \boldsymbol{S}^{(t)} + \tau\boldsymbol{M}^{(t)}$ and then follow by the orthogonalization described in step 5. In this case, step 5 is not optional anymore since $\boldsymbol{S}^{(t+1)}$ will move away from the Grassmannian after the Euclidean gradient update. The orthogonalization pulls $\boldsymbol{S}^{(t+1)}$ back to the Grassmann manifold. For ease of reference, we call this version of Riemannian SGD as *Alg. 1 variant*. We compare the two implementations in Tab. 7. The results show that the Grassmann class representation is effective on both versions of Riemannian SGD. We choose Alg. 1 because it is faster than the Alg. 1 variant. The thin SVD used in Equ. (3) can be efficiently computed via the *gesvda* approximate algorithm provided by the cuSOLVER library, which is faster than a QR decomposition on GPUs (see Tab. 9).

B. Details on Step 5 of Algorithm 1

The numerical inaccuracy is caused by the accumulation of tiny computational errors of Equ. (3). After running many iterations, the matrix $\boldsymbol{S}$ might not be perfectly orthogonal. For example, after 100, 1000, and 5000 iterations of the Grassmannian ResNet50-D with subspace dimension $k = 8$, we observed that the error $\max_i \|\boldsymbol{S}_i^T \boldsymbol{S}_i - \boldsymbol{I}\|_\infty$ is 1.9e-5, 9.6e-5 and 3.7e-4, respectively. After 50 epochs, the error accumulates to 0.0075. So, we run step 5 every 5 iterations to keep both the inaccuracies and the extra computational cost at a low level at the same time.

Table 7: Validation accuracy of Grassmann ResNet50-D on ImageNet with different retractions. The first row uses the exponential map, *i.e.,* moving along geodesics, as retraction, while the second row uses the Q factor of QR decomposition, *i.e.,* the qf function, as retraction.

Setting	Optimizer	Retraction	Top1	Top5
Alg. 1	RSGD+SGD	Geodesic	**79.26**	**94.44**
Alg. 1 Variant	RSGD+SGD	qf	79.13	94.45

Table 8: Validation accuracy of ResNet50-D on ImageNet trained with good initialization. Both rows use the weights of a ResNet50-D trained on ImageNet using the regular softmax. The first row fixes the backbone parameters, and solely learns the Grassmann fc layer, while the second row learns all parameters.

Setting	k	Initialization	Fine-Tune	Top1	Top5
GCR	8	Softmax pre-trained	Last layer	78.14	93.97
			All layers	**79.44**	**94.58**

C. The Importance of Joint Training

The joint training of the class subspaces and the features is essential. To support this claim, we add an experiment (first row of Tab. 8) that only fine-tunes the class subspaces from weights pre-trained using the regular softmax. We find that if the feature is fixed, changing the regular fc to the geometric version does not increase performance noticeably (top-1 from 78.04% of the regular softmax version to 78.14% of the Grassmann version). For comparison, we also add another experiment that fine-tunes all parameters (second row of Tab. 8). But when all parameters are free to learn, the pre-trained weights provide a good initialization that boosts the top-1 to 79.44%.

* Equal contribution.
† Corresponding author: Wayne Zhang.

Table 9: SVD and QR time (in *ms*) on Intel Xeon Gold 6146 (48) @ 3.201GHz and Nvidia V100 using PyTorch 1.13.1. The tested matrices are filled by Gaussian noises and have the shape of num classes $\times$ feature dimension $\times$ subspace dimension, which are the typical sizes encountered in Alg. 1 when training ImageNet-1K.

	SVD		**QR**	
Tensor Shape	CPU	GPU	CPU	GPU
$1000 \times 2048 \times 1$	5.1	20.0	2.6	36.5
$1000 \times 2048 \times 2$	11.3	20.7	7.8	46.0
$1000 \times 2048 \times 4$	59.9	21.2	18.8	56.3
$1000 \times 2048 \times 8$	138.0	23.2	78.8	70.9
$1000 \times 2048 \times 16$	352.9	25.2	200.5	118.5
$1000 \times 2048 \times 32$	1020.1	30.8	626.8	224.6

D. Influence on Training Speed

During training, the most costly operation in Alg. 1 is SVD. The time of SVD and QR on typical matrix sizes encountered in an iteration of Alg. 1 is benchmarked in Tab. 9. We can see that (1) SVD (with gesvda solver) is faster QR decomposition, and (2), when the subspace dimension is no greater than 2, CPU is faster than GPU. Based on these observations, in our implementation, we compute SVD on CPU when $k \leq 2$ and on GPU in other cases. Overall, when computing SVD, it adds roughly 5ms to 30ms overhead to the iteration time.

To measure the actual impact on training speed, we show the average iteration time (including a full forward pass and a full backward pass) of the vector class representation version *vs.* the Grassmann class representation version on different network architectures in Fig. 4. Overall, the Grassmann class representation adds about 0.3% (Deit3-S) to 35.0% (ResNet50-D) overhead. The larger the model, and the large the batch size, the smaller the relative computational cost.

E. More Visualizations on Principal Angles

Due to limited space, we only showed the visualization of the maximum and the minimum principal angles in Fig. 3. Here, we illustrate all eight principal angles in the GCR ($k = 8$) setting in Fig. 5.

F. Details on the Intra-Class Variability

In Section 5.2, we introduced the intra-class variability which is defined as the mean pairwise angles (in degrees) between features within the same class and then averaged over all classes. For models trained on the ImageNet-1K, we randomly sampled 200K training samples and use their global-centered feature to compute the intra-class variability. Suppose the set of global-centered features of class i is F_i,

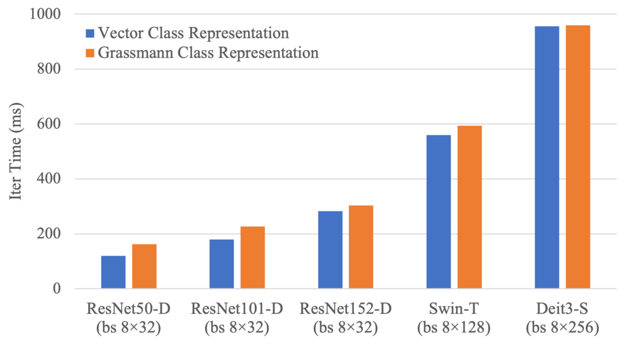

Figure 4: Compare the iteration time (in *ms*) between vector class representation and Grassmann class representation ($k = 8$) using different network architectures. Blue bars are networks with the original vector class representation and the orange bars are networks with the Grassmann class representation. The *bs 8×32* means that the batch size is 256, distributed across 8 GPUs and there are 32 samples per GPU.

Table 10: Details of the transfer datasets. The number of classes, the size of the training set and the testing set (or the validation set if no testing set or label of the testing set is not available), and the metric used to report the accuracies.

Dataset	Classes	Size (Train/Test)	Accuracy
CIFAR-10 [7],	10	50000/10000	top-1
CIFAR-100 [7],	10	50000/10000	top-1
Food-101 [3]	101	75750/25250	top-1
Oxford-IIIT Pets [10]	37	3680/3369	mean per-class
Stanford Cars [6]	196	8144/8041	top-1
Oxford 102 Flowers [9]	102	6552/818	mean per-class

then

$$\text{variability} := \frac{1}{C\,|F_i|^2} \sum_{i=1}^{C} \sum_{\boldsymbol{x}_j, \boldsymbol{x}_k \in F_i} \angle(\boldsymbol{x}_j, \boldsymbol{x}_k) \qquad (12)$$

where C is the number of classes, $\angle(\cdot, \cdot)$ is the angle (in degree) between two vectors, and $|F_i|$ is the cardinality of the set F_i.

G. Details on Transfer Datasets

In this section, we give the details of the datasets that are used in the feature transferability experiments. They are CIFAR-10 [7], CIFAR-100 [7], Food-101 [3], Oxford-IIIT Pets [10], Stanford Cars [6], and Oxford 102 Flowers [9]. The number of classes and the sizes of the training set and testing set are shown in Tab. 10.

H. Details on Linear SVM Hyperparameter

In Tab. 3, we used five-fold cross-validation on the training set to determine the regularization parameter of

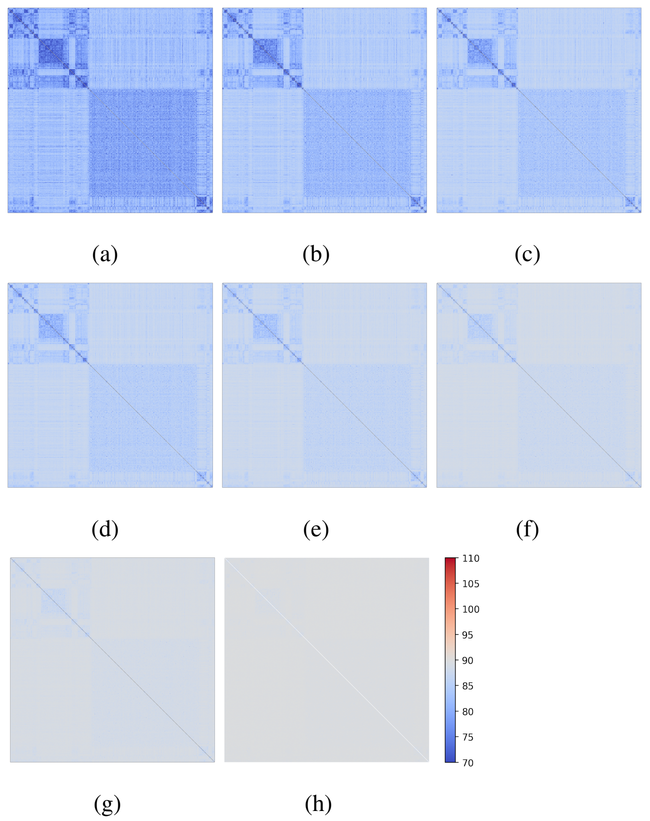

Figure 5: Each sub-figure is a heatmap of 1000×1000 grids. The color at the i-th row and the j-th column represent an angle between class i and class j in ImageNet-1K. Pairwise smallest (a)-(h) are pairwise principal angles between 8-dimensional class subspaces of a ResNet50-D model. (a) shows the smallest principal angels between any pair of classes; (b) shows the second smallest principal angles between any pair of classes; *etc.* Deeper Blue colors mean that their angles are smaller. Grayish colors mean the angles are close to 90°. Best viewed on screen with colors.

the linear SVM. The parameter is searched in the set $[0.1, 0.2, 0.5, 1, 2, 5, 10, 15, 20]$. Tab. 11 lists the selected regularization parameter of each setting. Both the cross-validation procedure and the SVM are implemented using the *sklearn* package. As a pre-processing step, the features are divided by the average norm of the respective training set, so that SVMs are easier to converge. The max iteration of SVM is set to 10,000.

I. Feature Transfer Using KNN

In Tab. 3, we have tested the feature transferability using linear SVM. Here we provide transfer results by KNN in Tab. 12. The hyperparameter K in KNN is determined by five-fold cross-validation on the training set. The candidate values are $1, 3, 5, \ldots, 49$. Our GCR demonstrates the best performance both on both CNNs and Vision Transformers.

Table 11: Regularization hyperparameter of SVM used in the linear feature transfer experiment. The hyperparameters are determined by five-fold cross-validation on the training sets. *C10* means CIFAR10 and *C100* means CIFAR100.

Setting		**Hyper-Parameter of SVM**					
Name	k	C10	C100	Food	Pets	Cars	Flowers
Softmax [4]		10	5	10	0.5	5	10
CosineSoftmax [5]		1	1	1	2	1	2
LabelSmoothing [12]		10	5	5	2	5	10
Dropout [11]		5	5	10	2	15	5
Sigmoid [2]		10	5	5	2	10	15
GCR (Ours)	1	1	0.5	0.5	1	1	2
	4	1	0.5	0.5	5	2	5
	8	1	0.5	0.5	1	1	2
	16	1	0.5	0.5	1	2	5
	32	1	1	0.5	2	2	2
Swin-T [8]		1	1	1	1	2	5
Swin-T GCR	8	0.5	1	1	2	2	10
Deit3-S [13]		2	2	2	2	5	10
Deit3-S GCR	8	1	1	0.5	0.5	2	2

For the ResNet50-D backbone, Grassmann with $k = 32$ has a better performance in both classification accuracy and transferability than all the baseline methods. On Swin-T, our method surpasses the original Swin-T by 2.76% on average. On Deit3-S, our method is 13.81% points better than the original Deit3-S. The experiments on KNN reinforced our conclusion that GCR improves large-scale classification accuracy and feature transferability simultaneously.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. Optimization algorithms on matrix manifolds. In *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[2] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014.

[4] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, pages 227–236. Springer, 1990.

[5] Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[6] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.

Table 12: Linear transfer using KNN for different losses and different backbones. All model weights are pre-trained on ImageNet-1K. In the first two sections, ResNet50-D is used as the backbone, and in the last section, Swin-T and Deit3-S are tested.

Setting		**Linear Transfer (KNN)**						
Name	k	CIFAR10	CIFAR100	Food	Pets	Cars	Flowers	**Avg.**
Softmax [4]		87.24	56.67	57.62	90.46	27.99	84.23	67.37
CosineSoftmax [5]		85.09	51.40	47.46	87.14	22.34	70.54	60.66
LabelSmoothing [12]		86.46	54.24	52.63	90.60	24.56	79.22	64.62
Dropout [11]		86.43	53.83	52.59	89.81	24.28	77.51	64.08
Sigmoid [2]		87.96	58.27	57.47	90.54	27.22	84.47	67.66
GCR (Ours)	1	86.65	52.72	46.83	86.73	21.58	66.99	60.25
	4	86.62	54.16	51.34	88.28	26.39	73.11	63.32
	8	86.84	55.64	53.31	87.93	27.97	79.22	65.15
	16	87.34	57.31	55.26	89.64	29.64	84.60	67.30
	32	86.96	56.39	56.88	89.75	30.31	87.04	67.89
Swin-T [8]		90.59	59.27	62.46	90.27	28.65	87.29	69.76
Swin-T GCR	8	91.38	62.04	66.57	91.91	32.40	90.83	72.52
Deit3-S [13]		86.04	50.54	45.47	88.12	18.22	63.69	58.68
Deit3-S GCR	8	91.64	63.80	65.18	91.80	33.88	88.63	72.49

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[9] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[10] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[13] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pages 516–533. Springer, 2022.