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

Haoqi Wang[2*] Zhizhong Li[1*] Wayne Zhang[13†]
[1] SenseTime Research [2] EPFL, Lausanne, Switzerland
[3] Guangdong Provincial Key Laboratory of Digital Grid Technology
haoqi.wang@epfl.ch {lizz,wayne.zhang}@sensetime.com

Abstract

We generalize the class vectors found in neural networks to linear subspaces (i.e., points in the Grassmann manifold) and show that the Grassmann Class Representation (GCR) enables simultaneous improvement in accuracy and feature transferability. In GCR, each class is a subspace, and the logit is defined as the norm of the projection of a feature onto the class subspace. We integrate Riemannian SGD into deep learning frameworks such that class subspaces in a Grassmannian are jointly optimized with the rest model parameters. Compared to the vector form, the representative capability of subspaces is more powerful. We show that on ImageNet-1K, the top-1 errors of ResNet50-D, ResNeXt50, Swin-T, and Deit3-S are reduced by 5.6%, 4.5%, 3.0%, and 3.5%, respectively. Subspaces also provide freedom for features to vary, and we observed that the intra-class feature variability grows when the subspace dimension increases. Consequently, we found the quality of GCR features is better for downstream tasks. For ResNet50-D, the average linear transfer accuracy across 6 datasets improves from 77.98% to 79.70% compared to the strong baseline of vanilla softmax. For Swin-T, it improves from 81.5% to 83.4% and for Deit3, it improves from 73.8% to 81.4%. With these encouraging results, we believe that more applications could benefit from the Grassmann class representation. Code is released at `https://github.com/innerlee/GCR`.

1. Introduction

The scheme `deep feature`$\rightarrow$`fully-connected` $\rightarrow$`softmax`$\rightarrow$`cross-entropy loss` has been the standard practice in deep classification networks. Columns of the weight parameter in the fully-connected layer are the class representative vectors and serve as the prototype for classes. The vector class representation has achieved huge success, yet it is not without imperfections. In the study of transferable features, researchers noticed a dilemma that representations with higher classification accuracy lead to less transferable features for downstream tasks [19]. This is connected to the fact that they tend to collapse intra-class variability of features, resulting in loss of information in the logits about the resemblances between instances of different classes [29]. The neural collapse phenomenon [34] indicates that as training progresses, the intra-class variation becomes negligible, and features collapse to their class means. As such, this dilemma inherently originates from the practice of representing classes by a single vector. This motivates us to study representing classes by high-dimensional subspaces.

Representing classes as subspaces in machine learning can be dated back, at least, to 1973 [49]. This core idea is re-emerging recently in various contexts such as clustering [54], few-shot classification [12, 41] and out-of-distribution detection [47], albeit in each case a different concrete instantiation was proposed. However, very few works study the subspace representation in large-scale classification, a fundamental computer vision task that benefits numerous downstream tasks. We propose the *Grassmann Class Representation* (GCR) to fill this gap and study its impact on classification and feature transferability via extensive experiments. To be specific, each class i is associated with a linear subspace S_i, and for any feature vector $\boldsymbol{x}$, the i-th logit l_i is defined as the norm of its projection onto the subspace S_i,

$$l_i := \left\|\operatorname{proj}_{S_i}\boldsymbol{x}\right\|. \tag{1}$$

In the following, we answer the two critical questions,

1. How to effectively optimize the subspaces in training?
2. Is Grassmann class representation useful?

Several drawbacks and important differences in previous works make their methodologies hard to generalize to the large-scale classification problem. Firstly, their subspaces might be not learnable. In ViM [47], DSN [41] and the SVD formulation of [54], subspaces are obtained *post hoc*

[*] Equal contribution. Work is done when Haoqi was at SenseTime.
[†] Corresponding author: Wayne Zhang.

by PCA-like operation on feature matrices without explicit parametrization and learning. Secondly, for works with learnable subspaces, their learning procedure for subspaces might not apply. For example, in RegressionNet [12], the loss involves *pairwise* subspace orthogonalization, which does not scale when the number of classes is large because the computational cost will soon be infeasible. And thirdly, the objective of [54] is unsupervised subspace clustering, which needs substantial changes to adapt to classification.

It is well known that the set of k-dimensional linear subspaces form a Grassmann manifold, so finding the optimal subspace representation for classes is to optimize on the Grassmannian. Therefore, a natural solution to Question 1 is to use geometric optimization [13], which optimizes the objective function under the constraint of a given manifold. Points being optimized are moving along geodesics instead of following the direction of Euclidean gradients. We implemented an efficient Riemannian SGD for optimization in the Grassmann manifold in Algorithm 1, which integrates the geometric optimization into deep learning frameworks so that the subspaces in Grassmannian and the model weights in Euclidean are jointly optimized.

The Grassmann class representation sheds light on the incompatibility issue between accuracy and transferability. Features can vary in a high-dimensional subspace without harming the accuracy. We empirically verify this speculation in Section 5, which involves both CNNs (ResNet [16], ResNet-D [17], ResNeXt [52], VGG13-BN [42]) and vision transformers (Swin [26] and Deit3 [45]). We found that with larger subspace dimensions k, the intra-class variation increase, and the feature transferability improve. The classification performance of GCR is also superior to the vector form. For example, on ImageNet-1K, the top-1 error rates of ResNet50-D, ResNeXt50, Swin-T and Deit3-S are reduced relatively by 5.6%, 4.5%, 3.0%, and 3.5%, respectively.

To summarize, our contributions are three folds. (1) We propose the Grassmann class representation and learn the subspaces jointly with other network parameters with the help of Riemannian SGD. (2) We showed its superior accuracy on large-scale classification both for CNNs and vision transformers. (3) We showed that features learned by the Grassmann class representation have better transferability.

2. Related Work

Geometric Optimization [13] developed the geometric Newton and conjugate gradient algorithms on the Grassmann and Stiefel manifolds in their seminal paper. Riemannian SGD was introduced in [6] with an analysis on convergence and there are variants such as Riemannian SGD with momentum [40] or adaptive [18]. Other popular Euclidean optimization methods such as Adam are also studied in the Riemannian manifold context [4]. [23] study the special case of $\mathrm{SO}(n)$ and $U(n)$ and uses the exponential map to enable Euclidean optimization methods for Lie groups. The idea was generalized into trivialization in [22]. Our Riemannian SGD Algorithm 1 is tailored for Grassmannian, so we use the closed-form equation for geodesics. Applications of geometric optimization include matrix completion [27, 25, 24, 32], hyperbolic taxonomy embedding [30], to name a few. [14] proposed the Grassmann discriminant analysis, in which features are modeled as linear subspaces.

Orthogonal Constraints Geometric optimization in deep learning is mainly used for providing orthogonal constraints in the design of network structure [15, 33], aiming to mitigate the gradient vanishing or exploding problems. Orthogonality are also enforced via regularizations [2, 51, 3, 37, 48]. Contrastingly, we do not change the network structures, and focus ourselves on the subspace form of classes. SiNN [39] uses the Stiefel manifold to construct Mahalanobis distance matrices in Siamese networks to improve embeddings in metric learning. It does not have the concept of classes.

Improving Feature Diversity Our GCR favors the intra-class feature variation by providing a subspace to vary. There are other efforts to encourage feature diversity. SoftTriplet loss [38] and SubCenterArcFace [10] model each class as local clusters with several centers or sub-centers. [55] uses a global orthogonal regularization to drive local descriptors spread out in the features space. [53] proposes to learn low-dimensional structures from the maximal coding rate reduction principle. The subspaces are estimated using PCA on feature vectors after the training.

Classes as Subspaces ViM [47] uses a subspace to denote the out-of-distribution class, which is obtained via PCA-like postprocessing after training. kSCN [54] uses subspaces to model clusters in unsupervised learning. Parameters of models and subspaces are optimized alternatively in a wake-and-sleep fashion. CosineSoftmax [19] defines logits via the inner product between the feature and normalized class vector. Since the class vector is normalized to be unit length, it is regarded as representing the class as a 1-dimensional subspace. ArcFace [11] improves over cosine softmax by adding angular margins to the loss. RegressionNet [12] uses the subspace spanned by the K feature vectors of each class in the N-way K-shot classification. The computational cost of its pairwise subspace orthogonalization loss is quadratic *w.r.t.* the number of classes and becomes infeasible when the number of classes is large. DSN [41] for few-shot learning computed subspaces from the data matrix rather than parametrized and learned, and its loss also involves pairwise class comparison which does not scale. Different from these formulations, we explicitly parametrize classes as high-dimensional subspaces and use geometric optimization to learn them in supervised learning.

3. Preliminaries

In this section, we briefly review the essential concepts in geometric optimization. Detailed exposition can be found in [13, 1]. Given an n-dimensional Euclidean space $\mathbb{R}^n$, the set of k-dimensional linear subspaces forms the Grassmann manifold $\mathcal{G}(k,n)$. A computational-friendly representation for subspace $S \in \mathcal{G}(k,n)$ is an orthonormal matrix $\boldsymbol{S} \in \mathbb{R}^{n\times k}$, where $\boldsymbol{S}^T\boldsymbol{S} = \boldsymbol{I}_k$ and $\boldsymbol{I}_k$ is the $k \times k$ identity matrix. Columns of the matrix $\boldsymbol{S}$ can be interpreted as an orthonormal basis for the subspace S. The matrix form is *not unique*, as right multiplying an orthonormal matrix will produce a new matrix representing the same subspace. Formally, Grassmannian is a quotient space of the Stiefel manifold and the orthogonal group $\mathcal{G}(k,n) = \mathrm{St}(k,n)/\mathcal{O}(k)$, where $\mathrm{St}(k,n) = \{\boldsymbol{X} \in \mathbb{R}^{n\times k} | \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_k\}$ and $\mathcal{O}(k) = \{\boldsymbol{X} \in \mathbb{R}^{k\times k} | \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_k\}$. When the context is clear, we use the space S and one of its matrix forms $\boldsymbol{S}$ interchangeably.

Given a function $f : \mathcal{G}(k,n) \to \mathbb{R}$ defined on the Grassmann manifold, the Riemannian gradient of f at point $S \in \mathcal{G}(k,n)$ is given by [13, Equ. (2.70)],

$$\nabla f(\boldsymbol{S}) = f_{\boldsymbol{S}} - \boldsymbol{S}\boldsymbol{S}^T f_{\boldsymbol{S}}, \tag{2}$$

where $f_{\boldsymbol{S}}$ is the Euclidean gradient with elements $(f_{\boldsymbol{S}})_{ij} = \frac{\partial f}{\partial \boldsymbol{S}_{ij}}$. When performing gradient descend on the Grassmann manifold, and suppose the current point is $\boldsymbol{S}$ and the current Riemannian gradient is $\boldsymbol{G}$, then the next point is the endpoint of $\boldsymbol{S}$ moving along the geodesic toward the tangent $\boldsymbol{G}$ with step size t. The geodesic is computed by [13, Equ. (2.65)],

$$\boldsymbol{S}(t) = \left(\boldsymbol{S}\boldsymbol{V}\cos(t\boldsymbol{\Sigma}) + \boldsymbol{U}\sin(t\boldsymbol{\Sigma})\right)\boldsymbol{V}^T, \tag{3}$$

where $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{G}$ is the thin SVD of $\boldsymbol{G}$.

4. Learning Grassmann Class Representation

Denote the weight of the last fully-connected (fc) layer in a classification network by $\boldsymbol{W} \in \mathbb{R}^{n\times C}$ and the bias by $\boldsymbol{b} \in \mathbb{R}^C$, where n is the dimension of features and C is the number of classes. The i-th column vector $\boldsymbol{w}_i$ of $\boldsymbol{W}$ is called the i-th class representative vector. The i-th logit is computed as the inner product between a feature $\boldsymbol{x}$ and the class vector (and optionally offset by a bias b_i), namely $\boldsymbol{w}_i^T\boldsymbol{x} + b_i$. We extend this well-established formula to a multi-dimensional subspace form $l_i := \left\|\mathrm{proj}_{S_i}\boldsymbol{x}\right\|$ where $S_i \in \mathcal{G}(k,n)$ is a k-dimensional subspace in the n-dimensional feature space. We call S_i the i-th *class representative space*, or class space in short. Comparing the new logit to the standard one, the inner product of feature $\boldsymbol{x}$ with class vector is replaced by the norm of the subspace projection $\mathrm{proj}_{S_i}\boldsymbol{x}$ and the bias term is omitted. We found that normalizing features to a constant length γ improves training. Incorporating this, Equ. (1) becomes

$$l_i := \left\|\mathrm{proj}_{S_i}\frac{\gamma\boldsymbol{x}}{\|\boldsymbol{x}\|}\right\|. \tag{4}$$

We assume $\boldsymbol{x}$ has been properly normalized throughout this paper so that we can simply use Equ. (1) in the discussion. We call this formulation of classes and logits the *Grassmann Class Representation* (GCR).

The subspace class formulation requires two changes to an existing network. Firstly, the last fc layer is replaced by the *Grassmann fully-connected layer*, which transforms features to logits using Equ. (4). Details can be found in Section 4.1. Secondly, the optimizer is extended to process the new geometric layer, which is explained in Section 4.2. Ultimately, parameters of the geometric layer are optimized using Riemannian SGD, while other parameters are simultaneously optimized using SGD, AdamW, or Lamb, *etc*.

4.1. Grassmann Class Representation

Suppose for class $i \in \{1,2,\ldots,C\}$, its subspace representation is $S_i \in \mathcal{G}(k_i,n)$, where the dimension k_i is a hyperparameter and is fixed during training. The tuple of subspaces $(S_1, S_2, \ldots, S_C)$ will be optimized in the product space $\mathcal{G}(k_1,n)\times\mathcal{G}(k_2,n)\times\cdots\times\mathcal{G}(k_C,n)$. Denote a matrix instantiation of S_i as $\boldsymbol{S}_i \in \mathbb{R}^{n\times k}$, where the column vectors form an orthonormal basis of S_i, then we concatenate these matrices into a big matrix

$$\boldsymbol{S} = [\boldsymbol{S}_1\ \boldsymbol{S}_2\ \cdots\ \boldsymbol{S}_C] \in \mathbb{R}^{n\times(k_1+k_2+\cdots+k_C)}. \tag{5}$$

The matrix $\boldsymbol{S}$ consists of the parameters that are optimized numerically. For a feature $\boldsymbol{x}$, the product $\boldsymbol{S}_i^T\boldsymbol{x}$ gives the coordinate of $\mathrm{proj}_{S_i}\boldsymbol{x}$ under the orthonormal basis formed by the columns of $\boldsymbol{S}_i$. By definition in Equ. (1), the logit for class i and the (normalized) feature $\boldsymbol{x}$ is

$$l_i = \left\|\mathrm{proj}_{S_i}\boldsymbol{x}\right\| = \left\|\boldsymbol{S}_i^T\boldsymbol{x}\right\|. \tag{6}$$

Grassmann Fully-Connected Layer We implement the geometric fully-connected layer using the plain old fc layer. The shape of the weight $\boldsymbol{S}$ is $n \times (k_1 + k_2 + \cdots + k_C)$, as shown in Equ. (5). In the forward pass, the input feature is multiplied with the weight matrix to get a temporary vector $\boldsymbol{t} = \boldsymbol{S}^T\boldsymbol{x}$, then the first element of the output is the norm of the sub-vector $(t_1, \ldots, t_{k_1})$, and the second element of the output is the norm of $(t_{k_1+1}, t_{k_1+2}, \ldots, t_{k_1+k2})$, and so on. If all k_i's be the same value k, as in our experiments, then the computation can be conveniently paralleled in one batch using tensor computation libraries.

Parameter Initialization Each matrix instantiation of the subspace should be initialized as an orthonormal matrix. To be specific, each block $\boldsymbol{S}_i$ of the weight $\boldsymbol{S}$ in Equ. (5) is orthonormal, while the matrix $\boldsymbol{S}$ needs not be orthonormal. For each block $\boldsymbol{S}_i$, we first fill them with standard Gaussian noises and then use $\mathrm{qf}(\boldsymbol{S}_i)$, namely the Q factor of its QR decomposition, to transform it to an orthonormal matrix. The geometric optimization Algorithm 1 will ensure their orthonormality during training.

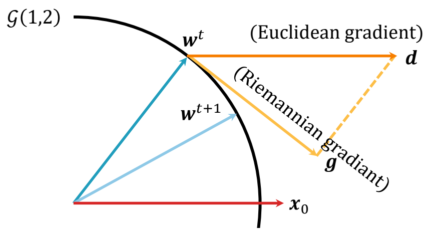

Figure 1: Geometric optimization in Grassmann manifold $\mathcal{G}(1,2)$. Each point (*e.g.* $\boldsymbol{w}^t$) in the black circle represent the 1-dimensional linear subspace S passing through it. The goal is to learn a subspace S to maximize $\|\mathrm{proj}_S \boldsymbol{x}_0\|$. $\boldsymbol{g}$ is the Riemannian gradient obtained by the projection of Euclidean gradient $\boldsymbol{d}$. $\boldsymbol{w}^t$ moves along the geodesic towards the direction $\boldsymbol{g}$ to a new point $\boldsymbol{w}^{t+1}$.

4.2. Optimize the Subspaces

Geometric optimization is to optimize functions defined on manifolds. The key is to find the Riemannian gradient *w.r.t.* the loss function and then descend along the geodesic. Here the manifold in concern is the Grassmannian $\mathcal{G}(k,n)$. As an intuitive example, $\mathcal{G}(1,2)$, composed of all lines passing through the origin in a two-dimensional plane, can be pictured as a unit circle where each point on it denotes the line passing through that point. Antipodal points represent the same line. To illustrate how geometric optimization works, we define a toy problem on $\mathcal{G}(1,2)$ that maximizes the norm of the projection of a fixed vector $\boldsymbol{x}_0$ onto a line through the origin, namely $\max_{S\in\mathcal{G}(1,2)} \|\mathrm{proj}_S \boldsymbol{x}_0\|$.

As shown in Fig. 1, we represent S with a unit vector $\boldsymbol{w} \in S$. Suppose at step t, the current point is $\boldsymbol{w}^{(t)}$, then it is easy to compute that the Euclidean gradient at $\boldsymbol{w}^{(t)}$ is $\boldsymbol{d} = \boldsymbol{x}_0$, and the Riemannian gradient $\boldsymbol{g}$ is the Euclidean gradient $\boldsymbol{d}$ projected to the tangent space of $\mathcal{G}(1,2)$ at point $\boldsymbol{w}^{(t)}$. The next iterative point $\boldsymbol{w}^{(t+1)}$ is to move $\boldsymbol{w}^{(t)}$ along the geodesic toward the direction $\boldsymbol{g}$. Without geometric optimization, the next iterative point would have lied at $\boldsymbol{w}^{(t)} + \gamma\boldsymbol{d}$, jumping outside of the manifold.

The following proposition computes the Riemannian gradient for the subspace in Equ. (1).

Proposition 1. *Let $\boldsymbol{S} \in \mathbb{R}^{n\times k}$ be a matrix instantiation of subspace $S \in \mathcal{G}(k,n)$, and $\boldsymbol{x} \in \mathbb{R}^n$ is a vector in Euclidean space, then the Riemannian gradient $\boldsymbol{G}$ of $l(S,\boldsymbol{x}) = \|\mathrm{proj}_S \boldsymbol{x}\|$ w.r.t. S is*

$$\boldsymbol{G} = \frac{1}{l}(\boldsymbol{I}_n - \boldsymbol{S}\boldsymbol{S}^T)\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{S}. \tag{7}$$

Proof. Rewrite $\|\mathrm{proj}_S \boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{x}}$, and compute the Euclidean derivatives as

$$\frac{\partial l}{\partial \boldsymbol{S}} = \frac{1}{l}\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{S}, \quad \frac{\partial l}{\partial \boldsymbol{x}} = \frac{1}{l}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{x}. \tag{8}$$

Then Equ. (7) follows from Equ. (2). □

Algorithm 1 An Iteration of the Riemannian SGD with Momentum for Grassmannian at Iteration t

```
Input: Learning rate $\tau > 0$, momentum $\mu \in [0,1)$, Grassmannian weight matrix $\boldsymbol{S}^{(t)} \in \mathbb{R}^{n\times k}$, momentum buffer $\boldsymbol{M}^{(t-1)} \in \mathbb{R}^{n\times k}$, Euclidean gradient $\boldsymbol{D} \in \mathbb{R}^{n\times k}$.
1: Riemannian gradient by Equ. (2), $\boldsymbol{G} \leftarrow (\boldsymbol{I}_n - \boldsymbol{S}\boldsymbol{S}^T)\boldsymbol{D}$.
2: Approximately parallel transport $\boldsymbol{M}$ to the tangent space of current point $\boldsymbol{S}^{(t)}$ by projection
       $\boldsymbol{M} \leftarrow (\boldsymbol{I}_n - \boldsymbol{S}\boldsymbol{S}^T)\boldsymbol{M}^{(t-1)}$.   (10)
3: Update momentum $\boldsymbol{M}^{(t)} \leftarrow \mu\boldsymbol{M} + \boldsymbol{G}$.
4: Move along geodesic using Equ. (3). If $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{M}^{(t)}$ is the thin SVD, then
       $\boldsymbol{S}^{(t+1)} \leftarrow \left(\boldsymbol{S}^{(t)}\boldsymbol{V}\cos(\tau\boldsymbol{\Sigma}) + \boldsymbol{U}\sin(\tau\boldsymbol{\Sigma})\right)\boldsymbol{V}^T$.
5: (Optional) Orthogonalization $\boldsymbol{S}^{(t+1)} \leftarrow \mathrm{qf}(\boldsymbol{S}^{(t+1)})$.
```

We give a geometric interpretation of Proposition 1. Let $\boldsymbol{w}_1$ be the unit vector along direction $\mathrm{proj}_S \boldsymbol{x}$, then expand it to an orthonormal basis of S, say $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k\}$. Since the Riemannian gradient is invariant to matrix instantiation, we can set $\boldsymbol{S} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \cdots \ \boldsymbol{w}_k]$. Then Equ. (7) becomes

$$\boldsymbol{G} = \left[\, (\boldsymbol{I}_n - \boldsymbol{S}\boldsymbol{S}^T)\boldsymbol{x} \ \ \boldsymbol{0} \ \ \cdots \ \ \boldsymbol{0} \,\right], \tag{9}$$

since $\boldsymbol{w}_i \perp \boldsymbol{x}, i = 2,3,\ldots,k$ and $\boldsymbol{w}_1^T\boldsymbol{x} = l$. Equ. (9) shows that in the single-sample case, only one basis vector $\boldsymbol{w}_1$, the unit vector in S that is closest to $\boldsymbol{x}$, needs to be rotated towards vector $\boldsymbol{x}$.

Riemannian SGD Parameters of non-geometric layers are optimized as usual using traditional optimizers such as SGD, AdamW, or Lamb during training. For the geometric Grassmann fc layer, its parameters are optimized using the Riemannian SGD (RSGD) algorithm. The pseudo-code of our implementation of RSGD with momentum is described in Algorithm 1. We only show the code for the single-sample, single Grassmannian case. It is trivial to extend them to the batch version and the product of Grassmannians. In step 2, we use projection to approximate the parallel translation of momentum, and the momentum update formula in step 3 is adapted from the official PyTorch implementation of SGD. Weight decay does not apply here since spaces are scaleless. Note that step 5 is optional since $\boldsymbol{S}^{(t+1)}$ in theory should be orthonormal. In practice, to suppress the accumulation of numerical inaccuracies, we do an extra orthogonalization step using $\mathrm{qf}(\cdot)$ every 5 iterations. Algorithm 1 works seamlessly with traditional Euclidean optimizers and converts the gradient from Euclidean to Riemannian on-the-fly for geometric parameters.

5. Experiment

In this section, we empirically study the influence of the Grassmann class representation under different settings. In Section 5.1, GCR demonstrates superior performance on the large-scale ImageNet-1K classification, a fundamental vision task. We experimented with both CNNs and vision transformers and observed consistent improvements. Then, in Section 5.2, we show that GCR improves the feature transferability by allowing larger intra-class variation. The choice of hyper-parameters and design decisions are studied in Section 5.3. Extra supportive experiments are presented in the supplementary material.

Experiment Settings For baseline methods, unless stated otherwise, we use the same training protocols (including the choice of batch size, learning rate policy, augmentation, optimizer, loss, and epochs) as in their respective papers. The input size is 224×224 for all experiments, and checkpoints with the best validation scores are used. All codes, including the implementation of our algorithm and re-implementations of the compared baselines, are implemented based on the *mmclassification* [28] package. PyTorch [36] is used as the training backend and each experiment is run on 8 NVIDIA Tesla V100 GPUs using distributed training.

Networks for the Grassmann class representation are set up by the drop-in replacement of the last linear fc layer in baseline networks with a Grassmann fc layer. The training protocol is kept the same as the baseline whenever possible. One necessary exception is to enhance the optimizer (*e.g.*, SGD, AdamW or Lamb) with RSGD (*i.e.*, RSGD+SGD, RSGD+AdamW, RSGD+Lamb) to cope with Grassmannian layers. To reduce the number of hyper-parameters, we simply set the subspace dimension k to be the same for all classes and we use $k = 8$ throughout this section unless otherwise specified. Suppose the dimension of feature space is n, then the Grassmann fully-connected layer has the geometry of $\Pi_{i=1}^{1000}\mathcal{G}(8, n)$. For hyper-parameters, we set $\gamma = 25$. Experiments with varying k's can be found in Section 5.2 and experiments on tuning γ are discussed in Section 5.3.

5.1. Improvements on Classification Accuracy

We apply Grassmann class representation to the large-scale classification task. The widely used ImageNet-1K [9] dataset, containing 1.28M high-resolution training images and 50K validation images, is used to evaluate classification performances. Experiments are organized into three groups which support the following observations. (1) It has superior performance compared with different ways of representing classes. (2) Grassmannian improves accuracy on different network architectures, including CNNs and the latest vision transformers. (3) It also improves accuracy on different training strategies for the same architecture.

Table 1: Validation accuracy of ResNet50-D on ImageNet-1K using different class representations.

Setting	Top1	Top5	Class Representation
Softmax [8]	78.04	93.89	vector class representation
CosineSoftmax [19]	78.30	94.07	1-dim subspace
ArcFace [11]	76.66	92.98	1-dim subspace with margin
MultiFC	77.34	93.65	8 fc layers ensembled
SoftTriple [38]	75.55	92.62	8 centers weighted average
SubCenterArcFace [10]	77.10	93.51	8 centers with one activated
GCR (Ours)	**79.26**	**94.44**	8-dim subspace with RSGD

On Representing Classes In this group, we compare seven alternative ways to represent classes. (1) **Softmax** [8] is the plain old vector class representation using the fc layer to get logits. (2) **CosineSoftmax** [19] represents a class as a 1-dimensional subspace since the class vector is normalized to be unit length. We set the scale parameter to 25 and do not add a margin. (3) **ArcFace** [11] improves over cosine softmax by adding angular margins to the loss. The default setting ($s = 64, m = 0.5$) is used. (4) **MultiFC** is an ensemble of independent fc layers. Specifically, we add 8 fc heads to the network. These fc layers are trained side by side, and their losses are then averaged. When testing, the logits are first averaged, and then followed by softmax to output the ensembled prediction. (5) **SoftTriple** [38] models each class by 8 centers. The weighted average of logits computed from multiple class centers is used as the final logit. We use the recommended parameters ($\lambda = 20, \gamma = 0.1, \tau = 0.2$ and $\delta = 0.01$) from the paper. (6) **SubCenterArcFace** [10] improves over ArcFace by using K sub-centers for each class and in training only the center closest to a sample is activated. We set $K = 8$ and do not drop sub-centers or samples since ImageNet is relatively clean. (7) The last setting is our **GCR** with subspace dimension $k = 8$. For all seven settings ResNet50-D is used as the backbone network and all models are trained on ImageNet-1K using the same training strategy described in the second row of Tab. 2.

Results are listed in Tab. 1, from which we find that the Grassmann class representation is most effective. Compared with the vector class representation of vanilla softmax, the top-1 accuracy improves from 78.04% to 79.26%, which amounts to 5.6% relative error reduction. Compared with previous ways of 1-dimensional subspace representation, *i.e.* CosineSoftmax and ArcFace, our GCR improves the top-1 accuracy by 0.96% and 2.60%, respectively. Compared with the ensemble of multiple fc, the top-1 is improved by 1.92%. Interestingly, simply extending the class representation to multiple centers such as SoftTriple (75.55%) and SubCenterArcFace (77.10%) does not result in good performances when training from scratch on the ImageNet-1K dataset. SoftTriple was designed for fine-grained classification and

Table 2: Comparing Grassmann class representation ($k = 8$) with vector class representation on different architectures. Validation accuracy on ImageNet. n is the feature dimension, *BS* means batch size, *WarmCos* means using warm up together with the cosine learning rate decay. *CE* is cross-entropy, *LS* is label smoothing, and BCE is binary cross-entropy.

	Setting				**Vector Class Representation**				**Grassmann Class Representation** ($k = 8$)			
Architecture	n	BS	Epoch	Lr Policy	Loss	Optimizer	**Top1**	**Top5**	Loss	Optimizer	**Top1**	**Top5**
ResNet50 [16]	2048	256	100	Step	CE	SGD	76.58	93.05	CE	RSGD+SGD	**77.77**(↑1.19)	**93.67**(↑0.62)
ResNet50-D [17]	2048	256	100	Cosine	CE	SGD	78.04	93.89	CE	RSGD+SGD	**79.26**(↑1.22)	**94.44**(↑0.55)
ResNet101-D [17]	2048	256	100	Cosine	CE	SGD	79.32	94.62	CE	RSGD+SGD	**80.24**(↑0.92)	**94.95**(↑0.33)
ResNet152-D [17]	2048	256	100	Cosine	CE	SGD	80.00	95.02	CE	RSGD+SGD	**80.44**(↑0.44)	**95.21**(↑0.19)
ResNeXt50 [52]	2048	256	100	Cosine	CE	SGD	78.02	93.98	CE	RSGD+SGD	**79.00**(↑0.98)	**94.28**(↑0.30)
VGG13-BN [42]	4096	256	100	Step	CE	SGD	72.02	90.79	CE	RSGD+SGD	**73.40**(↑1.38)	**91.30**(↑0.51)
Swin-T [26]	768	1024	300	WarmCos	LS	AdamW	81.06	95.51	LS	RSGD+AdamW	**81.63**(↑0.57)	**95.77**(↑0.26)
Deit3-S [45]	384	2048	800	WarmCos	BCE	Lamb	81.53	95.21	CE	RSGD+Lamb	**82.18**(↑0.65)	**95.73**(↑0.52)

SubCenterArcFace was designed for face verification. Their strong performances in their intended domains do not naively generalize here. This substantiates that making the subspace formulation competitive is a non-trivial contribution.

On Different Architectures We apply Grassmann class representation to eight network architectures, including six CNNs (ResNet50 [16], ResNet50/101/152-D [17], ResNetXt50 [52], VGG13-BN [42]) and two transformers (Swin [26], Deit3 [45]). For each model, we replace the last fc layer with Grassmannian fc and compare performances before and after the change. Their training settings together with validation top-1 and top-5 accuracies are listed in Tab. 2. The results show that GCR is effective across different model architectures. For all architectures, the improvement on top-1 is in the range $0.44-1.38\%$. The improvement is consistent not only for different architectures, but also across different optimizers (*e.g.*, SGD, AdamW, Lamb) and different feature space dimensions (*e.g.*, 2048 for ResNet, 768 for Swin, and 384 for Deit3).

On Different Training Strategies In this group, we train ResNet50-D with the three training strategies (RSB-A3, RSB-A2, and RSB-A1) proposed in [50], which aim to push the performance of ResNets to the extreme. Firstly, we train ResNet50-D with the original vector class representation and get top-1 accuracies of 79.36%, 80.29%, and 80.53%, respectively. Then, we replace the last classification fc with the Grassmann class representation ($k = 8$), and their top-1 accuracies improve to 79.88%, 80.74%, and 81.00%, respectively. Finally, we add the FixRes [46] trick to the three strategies, namely training on 176×176 image resolution and when testing, first resize to 232×232 and then center crop to 224×224. We get further boost in top-1 which are 80.20%, 81.04% and 81.29%, respectively. Results are summarized in Fig. 2.

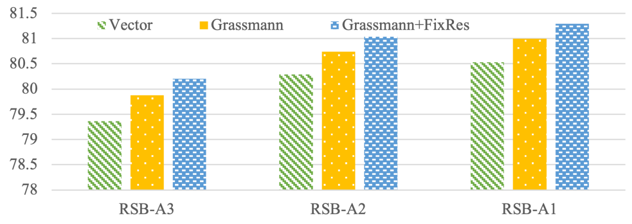

Figure 2: Validation accuracies of ResNet50-D on ImageNet-1K under different training strategies (RSB-A3, RSB-A2, and RSB-A1). Green bars are vector class representations; yellow bars are Grassmannian with $k = 8$; blue bars added the FixRes trick when training Grassmannian. The best top-1 of **ResNet50-D** is **81.29**%.

5.2. Improvements on Feature Transferability

In this section, we study the feature transferability of the Grassmann class representation. Following [19] on the study of better losses *vs.* feature transferability, we compare GCR with five different losses and regularizations. They are Softmax [8], Cosine Softmax [19], Label Smoothing [44] (with smooth value 0.1), Dropout [43] (with drop ratio 0.3), and the Sigmoid [5] binary cross-entropy loss. Note that baselines in Tab. 2 that do not demonstrate competitive classification performances are not listed here. The feature transfer benchmark dataset includes CIFAR-10 [21], CIFAR-100 [21], Food-101 [7], Oxford-IIIT Pets [35], Stanford Cars [20], and Oxford 102 Flowers [31]. All models are pre-trained on the ImageNet-1K dataset with the same training procedure as shown in the second row of Tab. 2. When testing on the transferred dataset, features (before the classification fc and Grassmann fc) of pre-trained networks are extracted. We fit linear SVMs with the one-vs-rest multi-class policy on each of the training sets and report their top-1 accuracies or mean class accuracies (for Pets and Flowers) on their test set. The regularization parameter for SVM is grid searched with candidates $[0.1, 0.2, 0.5, 1, 2, 5, 10, 15, 20]$ and determined by five-fold cross-validation on the training set.

Table 3: Linear transfer using SVM for different losses. ResNet50-D is used as the backbone, and model weights are pre-trained on ImageNet-1K. *Variability* measures the intra-class variability, and R^2 measures class separation.

Setting		ImageNet		Analysis		Linear Transfer (SVM)						
Name	k	Top-1	Top-5	Variability	R^2	CIFAR10	CIFAR100	Food	Pets	Cars	Flowers	**Avg.**
Softmax [8]		78.04	93.89	60.12	0.495	90.79	67.76	72.13	92.49	51.55	93.17	77.98
CosineSoftmax [19]		78.30	94.07	56.87	0.528	89.34	65.32	64.79	91.68	43.92	87.28	73.72
LabelSmoothing [44]		78.07	94.10	54.79	0.577	89.14	63.22	66.02	91.72	43.58	91.01	74.12
Dropout [43]		77.92	93.80	55.40	0.565	89.27	64.33	66.74	91.38	43.99	88.59	74.05
Sigmoid [5]		78.04	93.81	60.20	0.491	91.09	69.26	71.71	91.98	51.75	92.86	78.11
GCR (Ours)	1	78.42	94.14	56.50	0.534	89.98	66.34	64.34	91.37	42.97	86.85	73.64
	4	78.68	94.32	61.48	0.459	90.56	67.45	67.58	91.37	50.24	90.08	76.21
	8	**79.26**	**94.44**	63.49	0.430	90.13	67.90	70.06	91.85	53.25	92.64	77.64
	16	79.21	94.37	65.79	0.395	91.09	69.58	71.28	91.99	55.93	93.80	78.95
	32	78.63	94.05	67.74	0.365	91.35	69.49	71.80	92.47	58.05	95.04	**79.70**

Table 4: Feature transfer using Swin-T and Deit3-S. All model weights are pre-trained on ImageNet-1K as in Tab. 2. *C10/100* is CIFAR10/100, *Flwr* is Flowers. *Swin-T GCR* and *Deit3-S GCR* are their Grassmann variants.

Setting	Analysis		Linear Transfer (SVM)						
Architecture	Vari.	R^2	C10	C100	Food	Pets	Cars	Flwr	**Avg.**
Swin-T	60.2	0.48	92.7	69.4	77.5	92.1	61.3	96.0	81.5
Swin-T GCR	62.9	0.40	93.5	71.5	79.8	93.3	65.5	97.0	**83.4**
Deit3-S	50.6	0.60	89.5	63.7	64.7	91.4	43.1	90.2	73.8
Deit3-S GCR	61.5	0.44	93.0	71.9	74.9	92.3	60.7	95.5	**81.4**

Results The validation accuracies of different models on ImageNet-1K are listed in the second group of columns in Tab. 3. All GCR models ($k = 1, 4, 8, 16, 32$) achieve higher top-1 and top-5 accuracies than all the baseline methods with different losses or regularizations. Within a suitable range, a larger subspace dimension k improves the accuracy greater. However, when the subspace dimension is beyond 16, the top-1 accuracy begins to decrease. When $k = 32$, the top-1 is 78.63%, which is still 0.33% higher than the best classification baseline CosineSoftmax.

The linear transfer results are listed in the fourth group of columns in Tab. 3. Among the baseline methods, we find that Softmax and Sigmoid have the highest average linear transfer accuracies, which are 77.98% and 78.11%, respectively. Other losses demonstrate worse transfer performance than Softmax. For the Grassmann class representation, we observe a monotonic increase in average transfer accuracy when k increases from 1 to 32. When $k = 1$, the cosine softmax and the GCR have both comparable classification accuracies and comparable transfer performance. This can attribute to their resemblances in the formula. The transfer accuracy of GCR (73.64%) is lower than Softmax (77.98%) at this stage. Nevertheless, when the subspace dimension k increases, the linear transfer accuracy gradually improves, and when $k = 8$, the transfer performance (77.64%) is on par with the Softmax. When $k \geq 16$, the transfer performance surpasses all the baselines.

In Tab. 4, we show that features of the GCR version of Swin-T and Deit3 increase the average transfer accuracy by 1.9% and 7.6%, respectively.

Intra-Class Variability Increases with Dimension The intra-class variability is measured by first computing the mean pairwise angles (in degrees) between features within the same class and then averaging over classes. Following the convention in the study of neural collapse [34], the global-centered training features are used. [19] showed that alternative objectives which may improve accuracy over Softmax by collapsing the intra-class variability (see the *Variability* column in Tab. 3), degrade the quality of features on downstream tasks. Except for the Sigmoid, which has a similar intra-class variability (60.20) to Softmax (60.12), all other losses, including CosineSoftmax, LabelSmoothing, and Dropout, have smaller feature variability within classes (in the range from 54.79 to 56.87). However, the above conclusion does not apply when the classes are modeled by subspaces. For Grassmann class representation, we observed that if k is not extremely large, then *as k increases, both the top-1 accuracy and the intra-class variability grow.* This indicates that representing classes as subspaces enables the simultaneous improvement of inter-class discriminability and intra-class variability.

This observation is also in line with the class separation index R^2. R^2 is defined as one minus the ratio of the average intra-class cosine distance to the overall average cosine distance [19, Eq. (11)]. [19] founds that greater class separation R^2 is associated with less transferable features. Tab. 3 shows that when k increases, the class separation monotonically decreases, and the transfer performance grows accordingly.

Table 5: Validation accuracy of Grassmann ResNet50-D on ImageNet-1K with varying γ.

Setting	k	γ	Top1	Top5
ResNet50-D GCR	8	20	79.11	94.29
		25	**79.26**	**94.44**
		30	78.47	94.07

Table 6: Validation accuracy of Grassmann ResNet50-D on ImageNet with/without feature normalization.

Setting	k	Feature Normalize	Top1	Top5
ResNet50-D GCR	1		77.91	93.78
		✓	**78.42**	**94.14**
ResNet50-D GCR	8		78.12	93.90
		✓	**79.26**	**94.44**

5.3. Design Choices and Analyses

In this section, we use experiments to support our design choices and provide visualizations for the principal angles between class representative spaces.

Choice of Gamma In Tab. 5, we give more results with different values of γ when subspace dimension $k = 8$. We find $\gamma = 25$ has good performance and use it throughout the paper without further tuning.

Importance of Normalizing Features Normalizing the feature in Equ. (4) is critical to the effective learning of the Grassmann class representations. In Tab. 6 we compare results with/without feature normalization and observed a significant performance drop without normalization.

Principal Angles Between Class Representative Spaces When classes are subspaces, relationships between classes can be measured by k *principal angles*, which contain richer information than a single angle between two class vectors. The principal angles between two k-dimensional subspaces S and R are recursively defined as,

$$\begin{aligned} \cos(\theta_i) &= \max_{\boldsymbol{s}\in S}\max_{\boldsymbol{r}\in R} \boldsymbol{s}^T\boldsymbol{r} = \boldsymbol{s}_i^T\boldsymbol{r}_i, \\ s.t. \|\boldsymbol{s}\| &= \|\boldsymbol{r}\| = 1, \boldsymbol{s}^T\boldsymbol{s}_j = \boldsymbol{r}^T\boldsymbol{r}_j = 0, j \le i-1, \end{aligned} \tag{11}$$

for $i = 1, \ldots, k$ and $\theta_i \in [0, \pi/2]$. In Fig. 3, we illustrate the smallest and largest principal angles between any pair of classes for a model with $k = 8$. From the figure, we can see that the smallest principal angle reflects class similarity, and the largest principal angle is around $\pi/2$. A smaller angle means the two classes are correlated in some direction, and a $\pi/2$ angle means that some directions in one class subspace are completely irrelevant (orthogonal) to the other class.

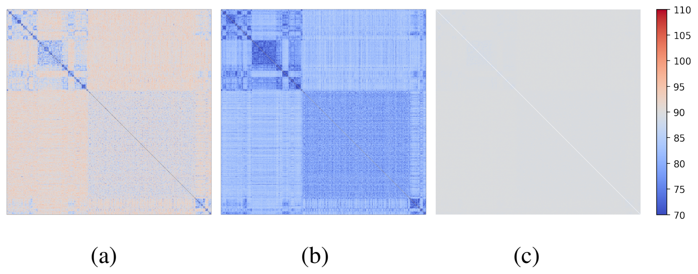

Figure 3: Each sub-figure is a heatmap of 1000×1000 grids. The color at the i-th row and the j-th column represent an angle between class i and class j in ImageNet-1K. (a) Pairwise angles between class vectors of the ResNet50-D trained by vanilla softmax. Grids with red hue is large than 90°, and blue hue means smaller than 90°. (b) Pairwise smallest principal angles between 8-dimensional class subspaces of a ResNet50-D model. Deeper blue colors indicate smaller angles. (c) Pairwise largest principal angles of the same model as in (b). Grayish color means they are close to 90°. Best viewed on screen with colors.

Necessity of Geometric Optimization To investigate the necessity of constraining the subspace parameters to lie in the Grassmannian, we replace the Riemannian SGD with the vanilla SGD and compare it with Riemannian SGD. Note that with SGD, the logit formula $\|\boldsymbol{S}_i^T x\|$ no longer means the projection norm because $\boldsymbol{S}_i$ is not guaranteed to be orthonormal anymore. With vanilla SGD, we get top-1 78.55% and top-5 94.18% when $k = 8$. The top-1 is 0.71% lower than models trained by Riemannian SGD.

6. Limitation and Future Direction

Firstly, a problem that remains open is how to choose the optimal dimension. Currently, we treat it as a hyper-parameter and decide it empirically. Secondly, we showed that the Grassmann class representation *allows for* greater intra-class variability. Given this, it is attractive to explore extensions to *explicitly promote* intra-class variability. For example, a promising approach is to combine it with self-supervised learning. We hope our work would stimulate progresses in this direction.

7. Conclusion

In this work, we proposed the Grassmann class representation as a drop-in replacement of the conventional vector class representation. Classes are represented as high-dimensional subspaces and the geometric structure of the corresponding Grassmann fully-connected layer is the product of Grassmannians. We optimize the subspaces using the optimization and provide an efficient Riemannian SGD implementation tailored for Grassmannians. Extensive experiments demonstrate that the new Grassmann class representation is able to improve classification accuracies on large-scale datasets and boost feature transfer performances at the same time.

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] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR, 2016.

[3] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? *Advances in Neural Information Processing Systems*, 31, 2018.

[4] Gary Becigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019.

[5] 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.

[6] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[7] 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.

[8] 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.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[10] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou. Sub-center arcface: Boosting face recognition by large-scale noisy web faces. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 741–757. Springer, 2020.

[11] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019.

[12] Arnout Devos and Matthias Grossglauser. Regression networks for meta-learning few-shot classification. In *7th ICML Workshop on Automated Machine Learning (AutoML 2020)*, number POST_TALK, 2020.

[13] Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[14] Jihun Hamm and Daniel D Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th international conference on Machine learning*, pages 376–383, 2008.

[15] Mehrtash Harandi and Basura Fernando. Generalized backpropagation, étude de cas: Orthogonality. *arXiv preprint arXiv:1611.05927*, 2016.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[18] Hiroyuki Kasai, Pratik Jawanpuria, and Bamdev Mishra. Riemannian adaptive stochastic gradient algorithms on matrix manifolds. In *International Conference on Machine Learning*, pages 3262–3271. PMLR, 2019.

[19] 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.

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

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

[22] Mario Lezcano Casado. Trivializations for gradient-based optimization on manifolds. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pages 3794–3803. PMLR, 2019.

[24] Zhizhong Li, Deli Zhao, Zhouchen Lin, and Edward Y. Chang. Determining step sizes in geometric optimization algorithms. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 1217–1221, 2015.

[25] Zhizhong Li, Deli Zhao, Zhouchen Lin, and Edward Y. Chang. A new retraction for accelerating the riemannian three-factor low-rank matrix completion algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[26] 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.

[27] Bamdev Mishra and Rodolphe Sepulchre. R3mc: A riemannian three-factor algorithm for low-rank matrix completion. In *53rd IEEE Conference on Decision and Control*, pages 1137–1142, 2014.

[28] MMClassification Contributors. OpenMMLab's Image Classification Toolbox and Benchmark, 7 2020.

[29] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[30] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018.

[31] 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.

[32] Madhav Nimishakavi, Pratik Kumar Jawanpuria, and Bamdev Mishra. A dual framework for low-rank tensor completion. *Advances in Neural Information Processing Systems*, 31, 2018.

[33] Mete Ozay and Takayuki Okatani. Training cnns with normalized kernels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[34] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[35] 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.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[37] Haozhi Qi, Chong You, Xiaolong Wang, Yi Ma, and Jitendra Malik. Deep isometric learning for visual recognition. In *International Conference on Machine Learning*, pages 7824–7835. PMLR, 2020.

[38] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6450–6458, 2019.

[39] Soumava Kumar Roy, Mehrtash Harandi, Richard Nock, and Richard Hartley. Siamese networks: The tale of two manifolds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3046–3055, 2019.

[40] Soumava Kumar Roy, Zakaria Mhammedi, and Mehrtash Harandi. Geometry aware constrained optimization techniques for deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4469, 2018.

[41] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4136–4145, 2020.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] 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.

[44] 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.

[45] 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.

[46] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *Advances in neural information processing systems*, 32, 2019.

[47] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4921–4930, 2022.

[48] Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11505–11515, 2020.

[49] Satosi Watanabe and Nikhil Pakvasa. Subspace method of pattern recognition. In *Proc. 1st. IJCPR*, pages 25–32, 1973.

[50] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*, 2021.

[51] Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6176–6185, 2017.

[52] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[53] Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning diverse and discriminative representations via the principle of maximal coding rate reduction. *Advances in Neural Information Processing Systems*, 33:9422–9434, 2020.

[54] Tong Zhang, Pan Ji, Mehrtash Harandi, Richard Hartley, and Ian Reid. Scalable deep k-subspace clustering. In *Asian Conference on Computer Vision*, pages 466–481. Springer, 2018.

[55] Xu Zhang, Felix X Yu, Sanjiv Kumar, and Shih-Fu Chang. Learning spread-out local feature descriptors. In *Proceedings of the IEEE international conference on computer vision*, pages 4595–4603, 2017.