Grassmann Pooling as Compact Homogeneous Bilinear Pooling for Fine-Grained Visual Classification

Xing Wei[1], Yue Zhang[1], Yihong Gong[1]*, Jiawei Zhang[2], and Nanning Zheng[1]

[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[2] SenseTime Research

Abstract. Designing discriminative and invariant features is the key to visual recognition. Recently, the bilinear pooled feature matrix of Convolutional Neural Network (CNN) has shown to achieve state-of-the-art performance on a range of fine-grained visual recognition tasks. The bilinear feature matrix collects second-order statistics and is closely related to the covariance matrix descriptor. However, the bilinear feature could suffer from the visual burstiness phenomenon similar to other visual representations such as VLAD and Fisher Vector. The reason is that the bilinear feature matrix is sensitive to the magnitudes and correlations of local CNN feature elements which can be measured by its singular values. On the other hand, the singular vectors are more invariant and reasonable to be adopted as the feature representation. Motivated by this point, we advocate an alternative pooling method which transforms the CNN feature matrix to an orthonormal matrix consists of its principal singular vectors. Geometrically, such orthonormal matrix lies on the Grassmann manifold, a Riemannian manifold whose points represent subspaces of the Euclidean space. Similarity measurement of images reduces to comparing the principal angles between these "homogeneous" subspaces and thus is independent of the magnitudes and correlations of local CNN activations. In particular, we demonstrate that the projection distance on the Grassmann manifold deduces a bilinear feature mapping without explicitly computing the bilinear feature matrix, which enables a very compact feature and classifier representation. Experimental results show that our method achieves an excellent balance of model complexity and accuracy on a variety of fine-grained image classification datasets.

Keywords: Fine-Grained Visual Classification; Bilinear Pooling; Singular Value Decomposition; Grassmann Manifold; Visual Burstiness

1 Introduction

Visual recognition problems mainly have two challenges: between-class similarity and within-class variance. Thus, designing discriminative and invariant features is the key to visual recognition [1–6]. The fine-grained image classification aims to

* Corresponding author

recognize subordinate categories of some base categories, such as different models of cars [7, 8], species of birds [9], variants of aircraft [10], kinds of foods [11], *etc.* Compared to general image classification, fine-grained classification is even more challenging since that visual differences between distinct fine-grained categories could be very small and subtle.

An approach to deal with such challenges is to incorporate strong supervision, for example, part-level and attribute annotations [12–14]. These methods first learn to detect semantic parts of the target object and then model the features of the local part for classification. Methods with strong supervision have shown to improve the fine-grained recognition accuracy significantly. However, annotating object parts is obviously much more expensive than assigning class labels. To avoid dependency on strong supervision, some have proposed to use the attention models [15–17] for unsupervised discovery of discriminative parts. Another promising approach is to absorb the effectiveness of training with web-scale datasets [18–20] via active learning.

Recently, a very simple method named bilinear pooling [21] has achieved state-of-the-art performance on a range of fine-grained classification benchmarks. The bilinear pooling method learns two separate CNNs whose outputs are multiplied using the outer product at each location and then summed to obtain a holistic representation of an image. The bilinear pooled matrix captures second-order statistics which is closely related to the covariance matrix descriptor [22].

A major drawback of the bilinear pooling is that the pooled feature is very high-dimensional. Thus the research line of this topic has focused on reducing the model complexity, both for the feature descriptor and classifier [23–25]. On the other hand, little attention has been paid to address the burstiness problem [26], where the feature elements may have large variances within the same class that adversely disturb the similarity measurement. Actually, bilinear pooling [21] and its variants [23–25] perform element-wise signed square root normalization to compensate for burstiness, taking the idea from other feature representations [26–28]. However, there is little analysis on how the bursts come into being in this framework.

Another approach [29] applies matrix power normalization where the singular values of the bilinear feature matrix are element-wise square rooted. Such normalization has shown to improve the performance of the bilinear feature. In fact, this idea is partially consistent with our opinion. We argue that the singular values are sensitive to the burstiness of visual elements while the singular vectors are more robust and reasonable to be considered as invariant features for recognition.

We thus advocate an alternative pooling method which transforms the CNN feature matrix to an orthonormal matrix consists of its principal singular vectors. Geometrically, such orthonormal matrix lies on the Grassmann manifold [30], a Riemannian manifold whose points represent subspaces of the Euclidean space. Similarity measurement of images reduces to comparing the principal angles between these "homogeneous" subspaces and thus is independent of the magnitudes and correlations of local CNN activations. Specifically, we show that the

projection distance [31, 32] of the Grassmann manifold deduces a bilinear feature mapping without explicitly computing the bilinear feature matrix, which leads to a very compact feature representation. Moreover, we also propose a Grassmann classifier that enjoys the same compact form which remarkably decreases the parameter size of the classifier. Finally, we propose a Grassmann projection approach in order to reduce the number of feature maps to compress our model further. Our previous work [33] adopts a triplet network to deal with the task of local patch matching. In this work, we focus on the basic image classification problem and analysis the connections to the bilinear pooling method in depth.

2 Related Work

Tenenbaum and Freeman [34] first introduced the bilinear model to separate style and content. Thereafter, second-order models have been studied in several computer vision problems, such as object detection [22], semantic segmentation [35], fine-grained classification [21], visual question answering [36], *etc.* Other feature representations have also been explored for visual recognition. The Bag-of-Words (BoW) framework [2, 3] used vector quantization for hard visual words assignment. While sparse coding [4, 5] improved by linear encoding with the sparse constraint. VLAD [37] and Fisher Vector [28] incorporated second-order information in the descriptors beyond linear encoding. The key issue to a feature representation is its discriminative and invariant power. Particularly, the burstiness problem has drawn much attention from various feature representations.

2.1 Bilinear Pooling and Variants

In this section, we briefly review several related bilinear pooling methods. The bilinear pooling [21] calculates second-order statistics of local features over the whole image to form a holistic representation for recognition. An obvious disadvantage of the original bilinear pooling is that pooled feature is very high-dimensional. To address this problem, Gao *et al.* [23] proposed two approximate methods via Random Maclaurin [38] and Tensor Sketch [39] to obtain compact bilinear representations. The compact models typically reduce the dimensionality up to 90% without losing noticeable classification accuracies. While the compact approximations reduce feature dimension remarkably, they ignore the matrix structure of the pooled feature matrix but vectorize it and apply a linear classifier. Kong *et al.* [24] proposed to maintain the matrix structure and learn a low-rank bilinear classifier. The resulting classifier can be evaluated without explicitly computing the bilinear feature matrix which allows a large reduction on the parameter size. Li *et al.* [25] proposed a similar idea to model pairwise feature interaction by performing a quadratic transformation with the low-rank constraint. They also proposed a regularization method to reduce the risk of over-fitting for the bilinear pooling. Lin and Maji [29] explored several matrix normalizations to increase the performance over the original bilinear feature. They found that the matrix power normalization outperforms several alternative schemes such as the matrix logarithm normalization.

2.2 On The Burstiness of Visual Representation

(a) Illumination and appearance changes (b) Repeated structures

Fig. 1. In general, visual burstiness corresponds to the problem that the feature representation is not invariant enough where the feature elements have large variances within the same class. The problem can be caused by (a) large illumination and appearance changes and (b) correlated elements such as repeated structures

The phenomenon of burstiness of visual elements was first explored in the BoW setting [26]: a given visual element appears many times in an image such that it can strongly affect the similarity measurement between two images since the contribution of other essential elements is substantially decreased. *Generally speaking, burstiness corresponds to the problem that the feature descriptor is not invariant enough where the feature elements may have large variances within the same class.* The problem can be caused by large illumination and appearance variations and correlated elements such as repeated structures, see Figure 1 for some examples. Visual burstiness has found widespread and important in many visual representations, from local patch descriptors to global image features.

Root-SIFT. Root-SIFT [40] transforms the original SIFT descriptor [1] by first L_1 normalizing the SIFT vector and then square rooting each element. It is shown that performing the scalar product in Root-SIFT space is equivalent to computing the Hellinger kernel in the original space. Because SIFT calculates the histogram of gradients, the effect of the Root-SIFT mapping can actually reduce the dominance of large gradient values and increase the importance of smaller but meaningful gradients.

Bag-of-Words (BoW). The BoW representation is obtained by quantizing the local descriptors into the visual words, resulting in frequency vectors. As noted by Jgou *et al.* [26], BoW can be very unbalanced caused by several high-frequent elements, usually as repeated patterns. This problem can be alleviated by discounting large values by element-wise square rooting the BoW vectors and re-normalizing them.

VLAD and Fisher Vector. In a similar manner, VLAD and Fisher Vector are signed square root normalized [37, 28]. To further suppress bursts, another kind of normalization termed intra-normalization was proposed in [27], where the sum of residuals is L_2 normalized within each VLAD block.

Bilinear Pooling. Similar to previous methods, the bilinear pooling method and its variants [21, 23, 24] also find that proper feature normalization provides a non-trivial improvement on performance. They consistently apply signed square root and L_2 normalization on the bilinear features. Another approach [29] compares several matrix based normalizations and find that matrix power normalization can improve the classification accuracy remarkably.

3 Grassmann Pooling as Compact Homogeneous Bilinear Pooling

To compute the bilinear feature matrix for an image, we first extract dense local image features by feeding it into a CNN. We take the output at a specific convolution layer, and form it as a matrix $\mathbf{A} \in \mathbb{R}^{c \times hw}$ where each row $i \in [0, c]$ represents the i'th feature map stacked to a 1D vector, and each column $j \in [0, hw]$ corresponds to a spatial location. The number, height, and width of feature maps are denoted by c, h, and w, respectively. Thus the symmetric form of bilinear pooling can be written in matrix notation by $\mathbf{B} = \mathbf{A}\mathbf{A}^T$.

3.1 Grassmann Pooling via Singular Value Decomposition

A major disadvantage of the bilinear pooling is that the produced feature is of high dimensionality. In the original bilinear pooling method [21], the pooled feature is reshaped into a vector $\mathbf{z} = vec(\mathbf{A}\mathbf{A}^T) \in \mathbb{R}^{c^2}$. Consider the VGG network which has $c = 512$ feature maps at the last convolution layer. Thus the dimensionality of the bilinear feature pooled at this layer is 2^{18}. Furthermore, if $c > hw$, thus $\mathbf{B} = \mathbf{A}\mathbf{A}^T$ is rank deficient. These reasons motivate us to find a more compact form of the bilinear feature matrix. To achieve this goal, we resort to Singular Value Decomposition (SVD) for low-rank matrix approximation. Before describing the pooling method, we first introduce two simple Lemmas.

Lemma 1. *Let* $\mathbf{A} = \sum_{i=1}^{c} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ *be the SVD of* $\mathbf{A}$ *and* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_c$. *For* $k \in \{1, 2, ..., c\}$, *let* $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ *be the sum truncated after* k *terms, thus* $\mathbf{A}_k$ *has rank* k. *We have, for any matrix* $\mathbf{X}$ *of rank at most* k, $\|\mathbf{A} - \mathbf{A}_k\|_F \leq \|\mathbf{A} - \mathbf{X}\|_F$, *where* $\|\cdot\|_F$ *is the Frobenius norm.*

Lemma 2. *Let* $\mathbf{A} = \sum_{i=1}^{c} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ *be the SVD of* $\mathbf{A}$, *if* $\mathbf{B} = \mathbf{A}\mathbf{A}^T$, *then*

$$\mathbf{B} = \left(\sum\nolimits_i \sigma_i \mathbf{u}_i \mathbf{v}_i^T\right)\left(\sum\nolimits_j \sigma_j \mathbf{u}_j \mathbf{v}_j^T\right)^T = \sum\nolimits_i \sum\nolimits_j \sigma_i \sigma_j \mathbf{u}_i \mathbf{v}_i^T \mathbf{v}_j \mathbf{u}_j^T = \sum\nolimits_i \sigma_i^2 \mathbf{u}_i \mathbf{u}_i^T \tag{1}$$

Lemma 1 shows that $\mathbf{A}_k$ is the best rank k approximation to $\mathbf{A}$ when error is measured by the Frobenius norm. Thus we can use SVD to find a low-rank approximation of the bilinear feature matrix without losing much accuracy. Lemma 2 gives two important information. First, the bilinear feature matrix $\mathbf{B}$ has the same singular vectors as the original feature matrix $\mathbf{A}$, and there is a one-to-one mapping between their singular values. So instead of approximating the bilinear feature matrix $\mathbf{B}$, we can just compute the SVD on the raw feature matrix $\mathbf{A}$, which could reduce computation complexity when $c > hw$. Second, the singular values of $\mathbf{B}$ change quadratically compared to those of $\mathbf{A}$. We argue that such phenomenon makes the bilinear feature matrix much more sensitive to the magnitudes and correlations of local CNN activations, which may cause the burstiness problem [26]. Consider that the original feature matrix $\mathbf{A}$ has a large singular value, thus it will be amplified dramatically in $\mathbf{B}$ and dominate the similarity measurement. To address this problem, we suggest the following pooling method.

Definition 1. *(Grassmann/Subspace Pooling) Let* $\mathbf{A} \in \mathbb{R}^{c\times hw}$ *be the feature maps at a specific convolution layer and* $\mathbf{A} = \sum_{i=1}^{c} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ *be the SVD, the Grassmann pooling or subspace pooling [33] reads:* $g_k(\mathbf{A}) = \mathbf{U}_k = [\mathbf{u}_1|\mathbf{u}_2|\cdots|\mathbf{u}_k]$.

That is, the pooling method transforms the CNN feature matrix $\mathbf{A}$ to an orthonormal matrix consists of its k principal left singular vectors. In geometry, the pooled CNN features obtained in this manner are k-dimensional linear subspaces of the c-dimensional Euclidean space, which lie on the (c, k) Grassmann manifold [30], denoted by $\mathcal{G}_c^k$. Now the bilinear feature matrix becomes $\mathbf{B}' = \mathbf{U}_k\mathbf{U}_k^T$. When inserted into a CNN and trained in an end-to-end fasion, this pooling method leads the model to learn only structural features which are independent of the magnitudes and correlations of visual elements. Please note that though singular values do not appear in this formulation, it does not mean that we think singular values are completely useless and discard all the information carried by them. Actually this representation can be understood as $\mathbf{B}' = \sum_{i=1}^{c} \sigma_i' \mathbf{u}_i \mathbf{v}_i^T$, where $\sigma_i' = 1$ for $i \in \{1, ..., k\}$ and $\sigma_i' = 0$ for $i \in \{k+1, ..., c\}$. We name bilinear pooling of the orthonormal matrix obtained in this manner as *homogeneous bilinear pooling*. Moreover, our method is also compact since $k < c$.

Figure 2 illustrates the differences of conventional bilinear pooling [21, 24], bilinear pooling with matrix power normalization [29] and our compact homogeneous bilinear pooling. Our pooling method has mainly two advantages. On the one hand, for large singular values that correspond to the major feature structures, our pooling method does not cause the burstiness problem since they are flattened to be ones. On the other hand, singular vectors correspond to the small singular values are often trivial structures or even noises, therefore are discarded by this representation which significantly reduces the feature size. Furthermore, we shall explain later it is even unnecessary to compute the homogenous bilinear feature matrix $\mathbf{B}' = \mathbf{U}_k\mathbf{U}_k^T \in \mathbb{R}^{c\times c}$ explicitly, but directly using the more

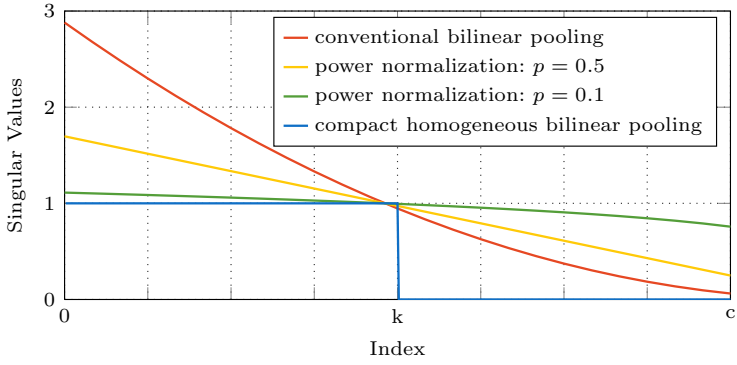


Fig. 2. A comparison of conventional bilinear pooling, bilinear pooling with matrix power normalization and our compact homogeneous bilinear pooling. Conventional bilinear pooling methods [21, 24] keep the whole spectrum and do not apply normalization on the singular values. Thus their feature matrices could suffer from the burstiness problem. The matrix power normalization [29] is proposed to handle this problem better. However, when the exponent p is approaching to zero, all the singular values are close to ones. Thus many trivial structures corresponding to the small singular values are amplified excessively. On the contrary, our compact homogeneous bilinear pooling adopts binary singular values, and its feature is compact which only contains the major feature structures corresponding to the large singular values and does not cause the burstiness problem

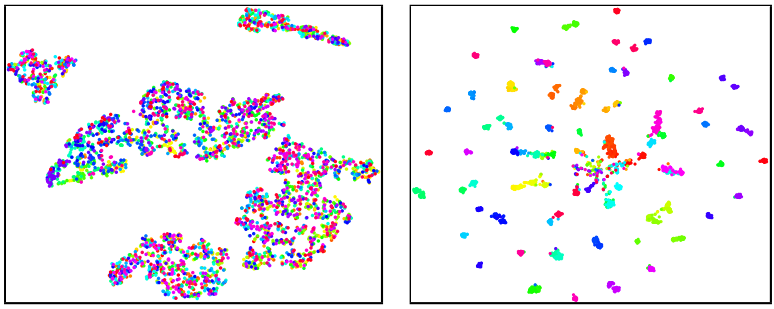
compact form $\mathbf{U}_k \in \mathbb{R}^{c\times k}$, where $k \ll c$ in practice [3]. The significance of this property resides in two respects. 1) For retrieval based or distributed applications which require features to be stored in a database, the storage can be substantially decreased. 2) For classification problems, especially when the numbers of categories is large, it can significantly reduce the parameter size of classifiers. Before explaining how to avoid the computation of bilinear feature matrix, we first make more analyses on the singular values and singular vectors.

3.2 Understanding Singular Values and Singular Vectors

To better understand the motivation of our pooling method, it is important to show some properties of singular values and singular vectors, respectively. We first consider two toy examples for analysis and then give some visualizations on a real dataset.

Toy Examples We consider two toy examples that mimic the burstiness phenomenon displayed in Figure 1 in a simplified way. This allows us to analyze the behaviors of singular values and singular vectors in a closed-form.

[3] For the rest of this paper, we use $\mathbf{U}$ instead of $\mathbf{U}_k$ for simplicity, and thereafter, the subscript will represent different examples, *e.g.*, $\mathbf{U}_1$ and $\mathbf{U}_2$.



(a) Singular values (b) Singular vectors

Fig. 3. Visualizing singular values and singular vectors as feature descriptors using t-SNE [41] using the original bilinear pooling method. (a) The singular values can be influenced by the magnitudes and correlations of local CNN feature elements, and they are wide-spread and mixed with different classes. (b) In contrast, the distributions of singular vectors are much more compact and easier to distinguished from each class

1. Linear illumination transform: $\mathbf{B} = (s\mathbf{A})(s\mathbf{A})^T = s^2\mathbf{A}\mathbf{A}^T = \sum_i s^2\sigma_i^2\mathbf{u}_i\mathbf{u}_i^T$. For a linear illumination transform when $\mathbf{A}$ is scaled by a scalar s, the singular values of $\mathbf{B}$ are scaled by s^2.
2. Repeated structures: $\mathbf{B} = [\mathbf{A}|\mathbf{A}]\,[\mathbf{A}|\mathbf{A}]^T = 2\mathbf{A}\mathbf{A}^T = \sum_i 2\sigma_i^2\mathbf{u}_i\mathbf{u}_i^T$. Consider that two duplicated feature matrix $\mathbf{A}$ generated by a CNN are concatenated together, thus the singular values of $\mathbf{B}$ are multiplied by a factor of 2.

As we can see from the two examples, singular values can actually reflect the magnitudes of the matrix and correlations on elements and could even be more sensitive to these factors when performing the bilinear pooling. On the other hand, singular vectors are more robust than singular values. They have unit norms, reflect only structural information and thus we believe should be more reasonable to consider as invariant features for recognition. Of course, real cases are much more complex than the toy examples where the illumination changes and correlations may not occur globally. Nevertheless, such examples show that how to pool CNN features in a more robust manner.

Real Cases We also make several visualizations on a real dataset. Specifically, we use the fine-grained aircraft dataset [10] as a testbed. We train the original bilinear pooling model on this dataset and extract the feature matrices of all the images in the test set. We then perform SVD on each feature matrix and get the singular values and singular vectors, respectively. We take $vec(\mathbf{U}\mathbf{U}^T)$ and $[\sigma_1, \sigma_2, \ldots]$ as two kinds of features and use the t-SNE [41] method to visualize them. As we can see from Figure 3(a), the singular values for each class spread widely and are mixed with different classes. On the contrary, the distributions of singular vectors are much more compact and easier to distinguish from each class as shown in Figure 3(b).

3.3 Learning Grassmann Classifiers

The Grassmann pooling method described in Section 3.1 maps each feature matrix to a subspace lies on the Grassmann manifold. The similarity measurement of images reduces to comparing the principal angles between these subspaces. For two points $\mathbf{U}_1$ and $\mathbf{U}_2$ on the $\mathcal{G}_c^k$ manifold, a popular distance measurement is the projection distance [31, 32] defined by $d_P(\mathbf{U}_1, \mathbf{U}_2) = k - \left\|\mathbf{U}_1^T\mathbf{U}_2\right\|_F^2$. Interestingly, we show that the projection distance deduces a bilinear feature mapping without explicitly computing the bilinear feature matrix, which leads to a very compact feature representation. Moreover, we can also train a Grassmann classifier that enjoys the same compact form thus the number of parameters of the classifier can be substantially reduced.

Lemma 3. *The projection distance deduces an implicit bilinear mapping of* $\mathbf{B}_1' = \mathbf{U}_1\mathbf{U}_1^T$ *and* $\mathbf{B}_2' = \mathbf{U}_2\mathbf{U}_2^T$:

$$\begin{aligned} d_P(\mathbf{U}_1, \mathbf{U}_2) &= k - \left\|\mathbf{U}_1^T\mathbf{U}_2\right\|_F^2 \\ &= \frac{1}{2}tr(\mathbf{U}_1\mathbf{U}_1^T) + \frac{1}{2}tr(\mathbf{U}_2\mathbf{U}_2^T) - tr(\mathbf{U}_1\mathbf{U}_1^T\mathbf{U}_2\mathbf{U}_2^T) \\ &= \frac{1}{2}tr(\mathbf{U}_1\mathbf{U}_1^T\mathbf{U}_1\mathbf{U}_1^T + \mathbf{U}_2\mathbf{U}_2^T\mathbf{U}_2\mathbf{U}_2^T - 2\mathbf{U}_1\mathbf{U}_1^T\mathbf{U}_2\mathbf{U}_2^T) \\ &= \frac{1}{2}\left\|\mathbf{U}_1\mathbf{U}_1^T - \mathbf{U}_2\mathbf{U}_2^T\right\|_F^2 \end{aligned} \tag{2}$$

Equation (2) indicates a valid similarity measurement between $\mathbf{U}_1$ and $\mathbf{U}_2$: $\left\|\mathbf{U}_1^T\mathbf{U}_2\right\|_F^2$, and we use this formula to define our classifier. For a K-way classification problem, we aim to learn K classifiers $\mathbf{W}_i \in \mathcal{G}_c^k, i \in [1, K]$. In particular, given an feature matrix $\mathbf{U} \in \mathcal{G}_c^k$, we compute a similarity score for each classifier by $\left\|\mathbf{W}_i^T\mathbf{U}\right\|_F^2$ and assign a label to the class which has the largest response. This formulation has a similar form to the bilinear SVM classifier defined in [24] but different in their meanings. The bilinear SVM [24] decomposes a classifier $\mathbf{W}_i \in \mathbb{R}^{c\times c}, rank(\mathbf{W}_i) = r, \mathbf{W}_i = \mathbf{W}_i^T$ into two parts:

$$\mathbf{W}_i = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{U}_i^T = \mathbf{U}_{i+}\mathbf{\Sigma}_{i+}\mathbf{U}_{i+}^T - \mathbf{U}_{i-}|\mathbf{\Sigma}_{i-}|\mathbf{U}_{i-}^T = \hat{\mathbf{U}}_{i+}\hat{\mathbf{U}}_{i+}^T - \hat{\mathbf{U}}_{i-}\hat{\mathbf{U}}_{i-}^T \tag{3}$$

The two parts are further relaxed to lie on the Euclidean space, *i.e.*, $\hat{\mathbf{U}}_{i-}$ and $\hat{\mathbf{U}}_{i+} \in \mathbb{R}^{c\times r/2}$. For an input feature matrix $\mathbf{X} \in \mathbb{R}^{c\times hw}$, the classification score is defined as $||\hat{\mathbf{U}}_{i+}^T\mathbf{X}||_F^2 - ||\hat{\mathbf{U}}_{i-}^T\mathbf{X}||_F^2$. On the contrary, our method is exactly derived from the projection distance on the Grassmann manifold.

To learn a Grassmann classifier $\mathbf{W}_i \in \mathcal{G}_c^k$, we first initialize a random matrix $\mathbf{M}_i \in \mathbb{R}^{c\times k}$ and then perform SVD on $\mathbf{M}_i$, assigning the left singular vectors to $\mathbf{W}_i$. Thus we train the classifiers end-to-end using the error back-propagation training method. Another better initialization of $\mathbf{W}_i$ is assigning each classifier to the center of its feature cluster. Specifically, for each class in the training set, the center is calculated by first summing all the features $\sum_j \mathbf{U}_j\mathbf{U}_j^T$, and then taking the singular vectors. We find that the later initialization facilitates CNN training and needs fewer epochs to convergence.

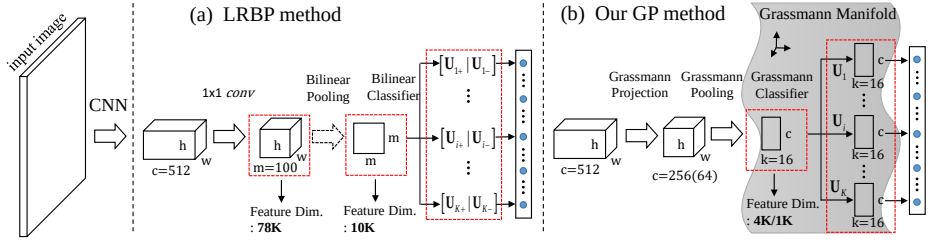


Fig. 4. Our GP method compares to the LRBP method [24]. Our GP mainly differs from LRBP in two respects: 1) pooling: GP transforms CNN features to the compact Grassmann manifold while LRBP uses the conventional (or does not explicitly compute) bilinear feature matrix. This directly matters on the feature dimension: 1K/4K vs. 10K/78K. 2) classifier: the classifier of GP is exactly derived from the projection distance of the Grassmann manifold; while for LRBP, it is derived from the bilinear SVM and is further approximated in the Euclidean space

3.4 Learning A Grassmann Projection for Model Compression

Typically, reducing the number of feature maps in CNN is performed by a 1×1 *conv* layer, setting a smaller number of the output feature maps. Mathematically, for a spatial location crosses all the channels of the feature maps, noted by a vector $\mathbf{x} \in \mathbb{R}^c$. The 1×1 *conv* layer learns a weight matrix $\mathbf{M} \in \mathbb{R}^{c\times c'}$ and gives the output $\mathbf{y} = \mathbf{M}^T\mathbf{x} \in \mathbb{R}^{c'}$ at each spatial location. This operation is equivalent to applying weighted sums along the channels of feature maps, and thus the outputs are linear combinations of the inputs. However, there may be correlations on these linear combinations such that the output feature maps are degenerate in terms of diversity.

To address this problem, we propose to learn c' ($c' < c$) orthonormal bases $\mathbf{W} \in \mathcal{G}_c^{c'}$ for dimension reduction. This approach is inspired by PCA which performs dimension reduction by mapping data using their principal orthonormal directions. The main difference between PCA and our method is that the proposed approach is supervised and trained end-to-end. To train the orthonormal directions, we first initialize a matrix $\mathbf{M}$, and set $\mathbf{W}$ to the left singular vectors of $\mathbf{M}$, similar to training the Grassmann classifiers. We will show in the experiment section that this dimension reduction approach performs favourably against the 1×1 *conv* operation.

The overall pipeline of our approach and a comparison with the LRBP method [24] is illustrated in Figure 4. Our method relies on SVD in several respects. The SVD operation is differentiable, thus the whole pipeline can be trained end-to-end using the standard back-propagation training procedure. However, computing the gradients with respect to SVD for back-propagation is non-trivial. Previously, Catalin *et al.* [42] have explored such matrix back-propagation in CNN which is also adopted in this work. We refer readers to [42] for the details of derivation.

4 Experimental Evaluation

In this section, we first provide details about the implementation of our method. We then compare our approach to several baselines on four widely used fine-grained classification benchmarks.

4.1 Implementation

We implement our method using the PyTorch library. We remove the fully connected layer of the original backbone network, and add our dimension reduction layer, pooling layer and classification layer. The stochastic gradient descent (SGD) optimization is used to train our model. We use an initial learning rate of 0.1 which linearly decreases to 0, weight decay of 0.0001 and momentum of 0.9. We perform random horizontal flipping as data augmentation for training. At the testing stage, we pass the original image and its horizontal flip to the CNN independently and use their average as the final classification score.

4.2 Baseline Methods

We now describe several baseline methods which we compare our approach with. To make a fair comparison, we choose the VGG-16 [43] network pre-trained on ImageNet as the backbone model which is the same as all the related bilinear methods. All the methods use the same K-way softmax loss function except for the low-rank bilinear pooling, which was specifically designed to incorporate a low-rank hinge loss [24].

Fully Connected layers (FC) [43]: This is the original VGG-16 network pre-trained on ImageNet. The last fully connected layer of VGG-16 is replaced with a randomly initialized K-way classification layer and then fine-tuned on each fine-grained classification dataset. Since VGG-16 requires a fixed input image size of 224×224, all input images are resized to this resolution for this method.

Full Bilinear Pooling (FBP) [21]: The full bilinear pooling is applied over the *conv5_3* feature maps, which is the best-performed B-CNN [D, D] in [21]. Before the final classification layer, an element-wise square root normalization and L_2 normalization layer is added as suggested in [21].

Compact Bilinear Pooling (CBP-RM and CBP-TS) [23]: We compare two compact methods proposed in [23] using Random Maclaurin [38] and Tensor Sketch [39]. The element-wise signed square root normalization and L_2 normalization are also used as the full bilinear model. The projection dimension is set to $d = 8192$, which is shown to be sufficient to achieve competitive performance as full bilinear pooling.

Low-Rank Bilinear Pooling (LRBP) [24]: This is the compression method using the bilinear SVM classifier. Following the original paper, the projection

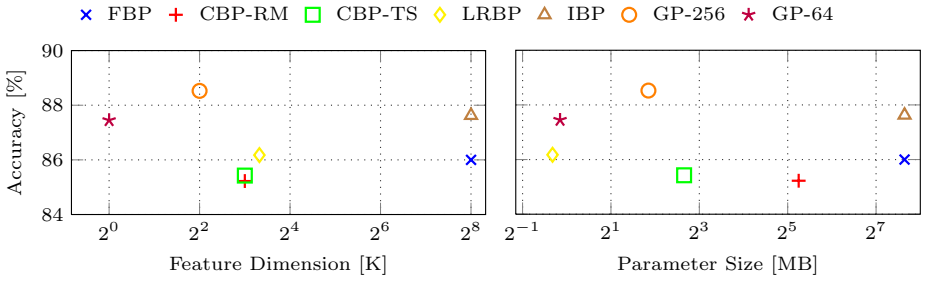


Fig. 5. A comparison of feature dimension/parameter size and accuracy for bilinear pooling methods. Accuracy is the average classification accuracy of the four datasets

dimension is set to $m = 100$ and its rank $r = 8$. The dimensionality reduction layer is initialized using PCA on the feature maps of training set, and a scaled square root is applied with a factor of 2×10^5 over the *conv*5_3 feature maps.

Improved Bilinear Pooling (IBP) [29]: This corresponds to the bilinear pooling with matrix square root normalization. Following the original paper, after fine-tuning the last layer is replaced by K one-vs-rest linear SVMs for classification.

4.3 Analysis of Model Complexity

In this section, we study the model and computational complexity for each method. For our Grassmann Pooling (GP) method, we use two different numbers of feature channels $c_1 = 256$ and $c_2 = 64$ (GP-256 and GP-64) and set $k = 16$ which is enough to achieve near maximum accuracy.

Table 1 provides a comprehensive comparison with respect to the feature dimension, the computational complexity for generating features and classification scores, and the parameter sizes of feature extractor (after the last convolution layer) and classifiers. For the CBP method, $d = 8192$ is used to achieve the best performance as reported in [23]. And for the LRBP method, $m = 100$ and $r = 8$ is enough to achieve similar or better performance than CBP. From Table 1 we can see that CBP-TS, LRBP, and our GP are most competitive with respect to model complexity. The feature dimension and parameter size are significantly smaller than FBP and IBP.

4.4 Performance Comparison on Benchmarks

We compare our method with related bilinear methods on four widely used fine-grained classification datasets, CUB Bird-200 [9], Stanford Car [7], Aircraft [10] and Food-101 [11]. All the datasets provide a fixed train and test split. We train our models only using the fine-grained class labels without any part or bounding box annotation provided by the datasets. The images are resized to 448×448 of all the datasets except for Food-101, which are resized to 224×224.

Table 1. A comparison of different bilinear pooling methods in terms of dimensionality, number of parameters, and computational complexity. The bilinear features are computed over feature maps of dimension $c \times h \times w$ for a K-way classification problem. For the VGG-16 [43] network on an input image of size 448×448, we have $c = 512$ and $h = w = 28$. FBP [21] and IBP [29] generate a full bilinear map which has a very high dimensionality. The CBP [23]: RM and TS use the polynomial kernel approximation, generating a feature vector of dimension d. Typically, the two methods can achieve near-maximum performance with $d = 8192$. LRBP [24] uses reduced feature dimension $m = 100$ and a low-rank classifier $r = 8$ to compress the bilinear model. We test two configurations of our method, setting $k = 16$ and $c_1 = 256, c_2 = 64$, respectively. Numbers in brackets indicate typical values when bilinear pooling is applied after the last convolutional layer of VGG-16 model over the CUB Bird-200 dataset [9] where $K = 200$. Model size only counts the parameters after the last convolutional layer

	Feature Dim.	Feature Comp.	Classifier Comp.	Feature Param.	Classifier Param.
FBP [21]	c^2 [256K]	$O(hwc^2)$	$O(Kc^2)$	0	Kc^2 [K·1MB]
CBP-RM [23]	d [8K]	$O(hwcd)$	$O(Kd)$	$2cd$ [40MB]	Kd [K·32KB]
CBP-TS [23]	d [8K]	$O(hw(c+d\log d))$	$O(Kd)$	$2c$ [4KB]	Kd [K·32KB]
LRBP [24]	m^2 [10K]	$O(hwmc+hwm^2)$	$O(Krm^2)$	cm [200KB]	Krm [K·3KB]
IBP [29]	c^2 [256K]	$O(hwc^2+c^3)$	$O(Kc^2)$	0	Kc^2 [K·1MB]
Our GP-256	kc_1 [4K]	$O(hwcc_1+c_1^3)$	$O(Kkc_1)$	cc_1 [512KB]	Kkc_1 [K·16KB]
Our GP-64	kc_2 [1K]	$O(hwcc_2+c_2^3)$	$O(Kkc_2)$	cc_2 [128KB]	Kkc_2 [K·4KB]

As can be seen from Table 2, all the bilinear pooling methods outperform the basic VGG-16 model a significant margin, especially for our GP-256 method which has the same feature dimension. The feature dimension of GP-256 is 64 times smaller than those of FBP and IBP with improved performance. LRBP has the smallest model among the baseline methods, while GP-64 outperforms it both in terms of accuracy and feature dimension with similar model size. The feature dimension of our GP-64 is only 10% of that of LRBP. An illustration of model size vs. classification accuracy for the bilinear pooling methods is displayed in Figure 5.

We also compare our Grassmann projection approach with the conventional 1×1 *conv* operation for dimension reduction. As can be seen from Table 3, the proposed Grassmann projection approach generally performs better than the 1×1 *conv*. It can reduce the number of feature maps by a factor of 2 of the original VGG-16 model, without losing noticeable classification accuracy.

5 Conclusions

This paper presents detailed analysis on the burstiness problem of the bilinear feature representation. We show that the bilinear feature matrix is sensitive to the magnitudes and correlations of local CNN feature elements which can be measured by its singular values. Thus we advocate an alternative pooling method

Table 2. Fine-grained image classification benchmark results. We compare our method with a fully connected network of VGG-16 [43], original bilinear pooling [21], Random Maclaurin and Tensor Sketch of the compact bilinear pooling [23], low-rank bilinear pooling [24], factorized bilinear pooling [25] and improved bilinear pooling [29] on four fine-grained classification datasets. We also list the feature dimension and parameter size for each method

	Bird [9]	Car [7]	Aircraft [10]	Food [11]	Feature Dim.	Parameters
FC [43]	70.4	76.8	74.1	80.9	4K	3MB
FBP [21]	84.1	90.6	86.9	82.4	256K	200MB
CBP-RM [23]	83.9	90.5	84.3	82.2	8K	38MB
CBP-TS [23]	84.0	91.2	84.1	82.4	8K	6.3MB
LRBP [24]	84.2	90.9	87.3	82.3	10K	0.8MB
IBP [29]	85.8	92.0	88.5	84.2	256K	200MB
Our GP-256	**85.8**	**92.8**	**89.8**	**85.7**	4K	3.6MB
Our GP-64	85.4	91.7	88.1	84.6	1K	0.9MB

Table 3. A comparison of dimension reduction methods. We compare our Grassmann projection approach with a 1×1 *conv* layer for reducing the number of feature maps. The results are reported using the proposed pooling method with $k = 16$ on the Aircraft [10] dataset. Values in the bracket are the numbers of feature maps and $1\times$ indicates the original VGG-16 model which has 512 feature maps

	$1\times$[512]	$2\times$[256]	$4\times$[128]	$8\times$[64]
1×1 *conv*	89.8	89.2	88.5	87.7
our method		89.8	88.9	88.1

that does not suffer from such a problem. Another advantage of our method is that it avoids to explicitly computing the bilinear feature matrix, which not only leads to a compact feature representation but also enables to train compact classifiers. Our method achieves an excellent balance of model complexity and accuracy on a variety of benchmarks for fine-grained image classification.

Acknowledgements

This work was supported by the National Basic Research Program of China (Grant No. 2015CB351705), and the State Key Program of National Natural Science Foundation of China (Grant No. 61332018).

References

1. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2) (2004) 91–110
2. Leung, T., Malik, J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. IJCV **43**(1) (2001) 29–44
3. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006) 2169–2178
4. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR. (2009) 1794–1801
5. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR. (2010) 3360–3367
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012) 1097–1105
7. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR). (2013)
8. Yang, L., Luo, P., Change Loy, C., Tang, X.: A large-scale car dataset for fine-grained categorization and verification. In: CVPR. (2015)
9. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset. (2011)
10. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013)
11. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: ECCV. (2014)
12. Zhang, N., Donahue, J., Girshick, R., Darrell, T.: Part-based r-cnns for fine-grained category detection. In: ECCV. (2014) 834–849
13. Zhang, X., Zhou, F., Lin, Y., Zhang, S.: Embedding label structures for fine-grained feature representation. In: CVPR. (2016)
14. Huang, S., Xu, Z., Tao, D., Zhang, Y.: Part-stacked cnn for fine-grained visual categorization. In: CVPR. (2016)
15. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NIPS. (2015) 2017–2025
16. Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., Zhang, Z.: The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In: CVPR. (2015)
17. Zheng, H., Fu, J., Mei, T., Luo, J.: Learning multi-attention convolutional neural network for fine-grained image recognition. In: ICCV. (2017)
18. Krause, J., Sapp, B., Howard, A., Zhou, H., Toshev, A., Duerig, T., Philbin, J., Fei-Fei, L.: The unreasonable effectiveness of noisy data for fine-grained recognition. In: ECCV. (2016) 301–320
19. Xu, Z., Huang, S., Zhang, Y., Tao, D.: Augmenting strong supervision using web data for fine-grained categorization. In: ICCV. (2015)
20. Cui, Y., Zhou, F., Lin, Y., Belongie, S.: Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In: CVPR. (2016)
21. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: ICCV. (2015)
22. Tuzel, O., Porikli, F., Meer, P.: Region covariance: A fast descriptor for detection and classification. In: ECCV. (2006) 589–600

23. Gao, Y., Beijbom, O., Zhang, N., Darrell, T.: Compact bilinear pooling. In: CVPR. (2016)
24. Kong, S., Fowlkes, C.: Low-rank bilinear pooling for fine-grained classification. In: CVPR. (2017)
25. Li, Y., Wang, N., Liu, J., Hou, X.: Factorized bilinear models for image recognition. In: ICCV. (2017)
26. Jégou, H., Douze, M., Schmid, C.: On the burstiness of visual elements. In: CVPR. (2009) 1169–1176
27. Arandjelovic, R., Zisserman, A.: All about vlad. In: CVPR. (2013)
28. Jegou, H., Perronnin, F., Douze, M., Sánchez, J., Perez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE TPAMI **34**(9) (2012) 1704–1716
29. Lin, T.Y., Maji, S.: Improved bilinear pooling with cnns. In: BMVC. (2017)
30. Turaga, P., Veeraraghavan, A., Chellappa, R.: Statistical analysis on stiefel and grassmann manifolds with applications in computer vision. In: CVPR. (2008) 1–8
31. Hamm, J., Lee, D.D.: Grassmann discriminant analysis: a unifying view on subspace-based learning. In: ICML. (2008)
32. Jayasumana, S., Hartley, R., Salzmann, M., Li, H., Harandi, M.: Kernel methods on riemannian manifolds with gaussian rbf kernels. IEEE TPAMI **37**(12) (2015) 2464–2477
33. Wei, X., Zhang, Y., Gong, Y., Zheng, N.: Kernelized subspace pooling for deep local descriptors. In: CVPR. (2018)
34. Tenenbaum, J.B., Freeman, W.T.: Separating style and content. In: NIPS. (1997) 662–668
35. Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C.: Semantic segmentation with second-order pooling. In: ECCV. (2012) 430–443
36. Fukui, A., Park, D.H., Yang, D., Rohrbach, A., Darrell, T., Rohrbach, M.: Multimodal compact bilinear pooling for visual question answering and visual grounding. arXiv preprint arXiv:1606.01847 (2016)
37. Jégou, H., Chum, O.: Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In: ECCV. (2012) 774–787
38. Kar, P., Karnick, H.: Random feature maps for dot product kernels. In: Artificial Intelligence and Statistics. (2012) 583–591
39. Pham, N., Pagh, R.: Fast and scalable polynomial kernels via explicit feature maps. In: ACM SIGKDD. (2013) 239–247
40. Arandjelović, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: CVPR. (2012) 2911–2918
41. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. JMLR **9** (2008) 2579–2605
42. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. In: ICCV. (2015)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)