Second-order Democratic Aggregation

Tsung-Yu Lin[1], Subhransu Maji[1], and Piotr Koniusz[2]

[1] College of Information and Computer Sciences
University of Massachusetts Amherst
{tsungyulin,smaji}@cs.umass.edu
[2] Data61/CSIRO, Australian National University
piotr.koniusz@data61.csiro.au

Abstract. Aggregated second-order features extracted from deep convolutional networks have been shown to be effective for texture generation, fine-grained recognition, material classification, and scene understanding. In this paper, we study a class of orderless aggregation functions designed to minimize *interference* or equalize *contributions* in the context of second-order features and we show that they can be computed just as efficiently as their first-order counterparts and they have favorable properties over aggregation by summation. Another line of work has shown that matrix power normalization after aggregation can significantly improve the generalization of second-order representations. We show that matrix power normalization implicitly equalizes contributions during aggregation thus establishing a connection between matrix normalization techniques and prior work on minimizing interference. Based on the analysis we present γ-democratic aggregators that interpolate between sum (γ=1) and democratic pooling (γ=0) outperforming both on several classification tasks. Moreover, unlike power normalization, the γ-democratic aggregations can be computed in a low dimensional space by sketching that allows the use of very high-dimensional second-order features. This results in a state-of-the-art performance on several datasets.

Keywords: Second-order features, democratic pooling, matrix power normalization, tensor sketching

1 Introduction

Second-order statistics have been demonstrated to improve performance of classification on images of objects, scenes and textures as well as fine-grained problems, action classification and tracking [52,44,30,15,6,25,34,33]. In the simplest form, such statistics are obtained by taking the outer product of some feature vectors and aggregating them over some region of interest which results in an auto-correlation [6,24] or covariance matrix [52]. Such a second-order image descriptor is then passed as a feature to train a SVM, *etc.* Several recent works obtained an increase in accuracy after switching from the first- to second-order statistics [25,24,34,32,33,59,26,28]. Further improvements were obtained by considering the impact of spectrum of such statistics on aggregation into the final representations [25,24,37,31,33,26,28]. For instance, analysis conducted in [25,24] concluded that decorrelating feature vectors from an image via the matrix power normalization has a positive impact on classification due to the signal whitening properties which prevent so-called *bursts* of features [19]. However, evaluating the power of matrix is a costly procedure with complexity $\mathcal{O}(d^{\omega})$, where $2<\omega<2.376$ concerns the complexity of SVD. In recent CNN approaches [31,33,28] which perform end-to-end learning, the complexity becomes a prohibitive factor for typical $d \geq 1024$ due to a costly backpropagation step which involves SVD or solving a Lyapunov equation [33] in every iteration of the CNN fine-tuning process; thus adding several hours of computations to training. However, another line of aggregation mechanisms aim to reweight the first-order feature vectors prior to their aggregation [37] in order to balance their contributions to the final image descriptor. Such a reweighting scheme, called a democratic aggregation [21,37], is solved very efficiently by a modified Sinkhorn algorithm [23].

In this paper, we study democratic aggregation in the context of second-order feature descriptors and show that this feature descriptor has favorable properties when combined with the democratic aggregator which was applied originally to the first-order descriptors. We take a closer look at the relation between the reweighted representations and the matrix power normalization in terms of the variance of feature contributions. In addition, we propose a γ-democratic aggregation scheme which generalizes democratic aggregation and allows to interpolate between sum pooling and democratic pooling. We show that our formulation can be solved via the Sinkhorn algorithm as efficiently as approach [37] while resulting in a performance comparable to the matrix power normalization. Computationally, our approach involves Sinkhorn iterations, which requires matrix-vector multiplications, and is faster by an order of magnitude even when compared to approximate matrix power normalization via the Newton's method, which involves matrix-matrix operations [33]. Unlike the power matrix normalization, our γ-democratic aggregation can be performed via sketching [42,12] enabling the use of high-dimensional feature vectors.

To summarize, our contributions are: (i) we propose a new second-order γ-democratic aggregation, (ii) we obtain reweighting factors via the Sinkhorn algorithm which enjoys an order of magnitude speedup over the fast matrix power normalization via Newton's iterations while it achieves comparable results, (iii) we provide theoretical bounds on feature contributions in relation to the matrix power normalization, (iv) we present state-of-the-art results on several datasets by applying democratic aggregation of second-order representations with sketching.

2 Related work

Mechanisms of aggregating first- and second-order features have been extensively studied in the context of image retrieval, texture and object recognition [40,41,47,20,38,44,53,6,25]. In what follows, we first describe shallow approaches and non-Euclidean aggregation schemes followed by the CNN-based approaches.

Shallow Approaches. Early approaches to aggregating second-order statistics include Region Covariance Descriptors [44,53], Fisher Vector Encoding [40,41,47] and Vector of Locally Aggregated Tensors [38], to name but a few of approaches.

Region Covariance Descriptors capture co-occurrences of luminance, first- and second-order partial derivatives of images [44,53] and, in some cases, even binary patterns [46]. The main principle of these approaches is to aggregate the co-occurrences of some feature vectors into a matrix which represents an image.

Fisher Vector Encoding [40] precomputes a visual vocabulary by clustering over a set of feature vectors and captures the element-wise squared difference between each feature vector and its nearest cluster center. Subsequently, the re-normalization of the captured statistics with respect to the cluster variance and the sum aggregation are performed. Furthermore, extension [41] proposes to apply the element-wise square root to the aggregated statistics which improves the classification results. Vector of Locally Aggregated Tensors extends Fisher Vector Encoding to second-order off-diagonal feature interactions.

Non-Euclidean Distances. To take the full advantage of statistics captured by the scatter matrices, several works employ non-Euclidean distances. For positive definite matrices, geodesic distances (or their approximations) known from the Riemannian geometry are used [39,3,2]. Power-Euclidean distance [11] extends to semidefinite positive matrices. Distances such as Affine-Invariant Riemannian Metric [39,3], KL-Divergence Metric [55], Jensen-Bregman LogDet Divergence [7] and Log-Euclidean distance [2] are frequently used for comparing scatter matrices resulting from aggregation of second-order statistics. However, the above distances are notoriously difficult to backpropagate through for end-to-end learning and often computationally prohibitive [27].

Pooling Normalizations. Both first- and second-order aggregation methods often employ normalizations of pooled feature vectors. The early works on image retrieval apply the square root [19] to aggregated feature vectors to limit the impact of frequently occurring features and boost the impact of infrequent and highly informative ones (so-called notion of feature *bursts*). The roots of this approach in computer vision can be traced back to so-called generalized histogram of intersection kernel [5]. For second-order approaches, similar strategy is used by Fisher Vector Encoding [41]. The notion of *bursts* is further studied in the context of Bags-of-Words approach as well scatter matrices and tensors for which their spectra are power normalized [24,25,26] (so-called Eigenvalue Power Normalization or EPN for short). However, the square complexity of scatter matrices w.r.t. length of feature vectors deems them somewhat impractical in classification. A recent study [21,37] shows how to exploit second-order image-wise statistics and reweight sets of feature vectors per image at the aggregation time to obtain an informative first-order representation. So-called Democratic Aggregation (DA) and Generalized Max-Pooling (GMP) strategies are proposed whose goal is to reweight feature vectors per image prior to the sum aggregation so that interference between frequent and infrequent feature vectors is minimized. Strategies such as EPN (Matrix Power Normalization, MPN for short, is a special case of

EPN), DA and GMP can be seen as ways of equalizing contributions of feature vectors into the final image descriptor and they are closely related to Zero-phase Component Analysis (ZCA) whose role is to whiten the signal representation.

Pooling and Aggregation in CNNs. The early image retrieval and recognition CNN-based approaches aggregate first-order statistics extracted from the CNN maps *e.g.*, [14,57,1]. In [14], multiple feture vectors are aggregated over multiple image regions. In [57], feature vectors are aggregated for retrieval. In [1], so-called VLAD descriptor is extended to allow end-to-end training.

More recent approaches form co-occurrence patterns from CNN feature vectors similar in spirit to Region Covariance Descriptors. In [34], the authors combine two CNN streams of feature vectors via outer product and demonstrate that such a setup is robust for the task of the fine-grained image recognition. A recent approach [49] extracts feature vectors at two separate locations in feature maps and performs an outer product to form a CNN co-occurrence layer.

Furthermore, a number of recent approaches are dedicated to performing backpropagation on the spectrum-normalized scatter matrices [18,17,31,33,28]. In [18], the authors employ the backpropagation via the SVD of matrix to implement the Log-Euclidean distance in end-to-end fashion. In [31], the authors extend Eigenvalue Power Normalization [25] to an end-to-end learning scenario which also requires to backpropagate via the SVD of matrix. Concurrently, approach [33] suggests to perform Matrix Power Normalization via the Newton's method and backpropagate w.r.t. the square root of matrix by solving a Lyapunov equation for greater numerical stability. An approach [58] phrases the matrix normalization as the problem of robust covariance estimation. Lastly, compact bilinear pooling [12] uses so-called tensor sketching [42]. Where indicated, we also make use of tensor sketching in our work.

There has been no connection made between reweighting feature vectors and its impact on the spectrum of the corresponding scatter matrix. Our work closely related to the approaches [21,37], however, introduce a mechanisms of limiting the interference in the context of second-order features. We demonstrate their superiority over the first-order inference approaches [21,37] and show that we can obtain results comparable to the matrix square root aggregation [33] with much lower computational complexity at the training and testing stages.

3 Method

Given a sequence of features $\mathcal{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, where $\mathbf{x}_i \in \mathbb{R}^d$, we are interested in a class of functions that compute an *orderless aggregation* of the sequence to obtain a global descriptor $\xi(\mathcal{X})$. If the descriptor is orderless, it implies that any permutation of features does not effect the global descriptor. A common approach is to encode each feature using a non-linear function $\phi(\mathbf{x})$ before aggregation via a simple symmetric function such as sum or max. For example, the global descriptor using sum pooling can be written as:

$$\xi(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \phi(\mathbf{x}). \tag{1}$$

In this work, we investigate outer-product encoders, *i.e.* $\phi(\mathbf{x}) = \text{vec}(\mathbf{x}\mathbf{x}^T)$, where $\mathbf{x}^T$ denotes the transpose and $\text{vec}(\cdot)$ is the vectorization operator. Thus, if $\mathbf{x}$ is d dimensional then $\phi(\mathbf{x})$ is d^2 dimensional.

3.1 Democratic aggregation

The democratic aggregation approach was proposed in [37] to minimize interference or equalize contributions of each element in the sequence. The contribution of a feature is measured as the similarity of the feature to the overall descriptor. In the case of sum pooling, the contribution $C(\mathbf{x})$ of a feature $\mathbf{x}$ is given by:

$$C(\mathbf{x}) = \phi(\mathbf{x})^T \sum_{\mathbf{x}' \in \mathcal{X}} \phi(\mathbf{x}'). \tag{2}$$

For sum pooling, the contributions $C(\mathbf{x})$ may not be equal for all features $\mathbf{x}$. In particular, the contribution is affected by both the norm and frequency of the feature. Democratic aggregation is a scheme that weights each feature by a scalar $\alpha(\mathbf{x})$ that depends on both $\mathbf{x}$ and the overall set of features in $\mathcal{X}$ such that the weighted aggregation $\xi(\mathcal{X})$ satisfies:

$$\alpha(\mathbf{x})\phi(\mathbf{x})^T\xi(\mathcal{X}) = \alpha(\mathbf{x})\phi(\mathbf{x})^T \sum_{\mathbf{x}' \in \mathcal{X}} \alpha(\mathbf{x}')\phi(\mathbf{x}') = C,\ \forall \mathbf{x} \in \mathcal{X}, \tag{3}$$

under the constraint that $\forall \mathbf{x} \in \mathcal{X}$, $\alpha(\mathbf{x}) > 0$. The above equation only depends on the dot product between the elements since:

$$\alpha(\mathbf{x}) \sum_{\mathbf{x}' \in \mathcal{X}} \alpha(\mathbf{x}')\phi(\mathbf{x})^T\phi(\mathbf{x}') = \alpha(\mathbf{x}) \sum_{\mathbf{x}' \in \mathcal{X}} \alpha(\mathbf{x}')k(\mathbf{x}, \mathbf{x}'), \tag{4}$$

where $k(\mathbf{x}, \mathbf{x}')$ denotes the dot product between the two vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$. Following the notation in [37], if we denote $\mathbf{K}_{\mathcal{X}}$ to be the kernel matrix of the set $\mathcal{X}$, the above constraint is equivalent to finding a vector of weights $\boldsymbol{\alpha}$ such that:

$$\texttt{diag}(\boldsymbol{\alpha})\mathbf{K}\texttt{diag}(\boldsymbol{\alpha})\mathbf{1}_n = C\mathbf{1}_n, \tag{5}$$

where `diag` is the diagonalization operator and $\mathbf{1}_n$ is an n dimensional vector of ones. In practice, the aggregated features $\xi(\mathcal{X})$ are ℓ_2 normalized hence the constant C does not matter and can be set to 1.

The authors [37] noted that the above equation can be efficiently solved by a dampened Sinkhorn algorithm [23]. The algorithm returns a unique solution as long as certain conditions are met, namely the entries in $\mathbf{K}$ are non-negative and the matrix is not fully decomposable. In practice, these conditions are not satisfied since the dot product between two features can be negative. A solution proposed in [37] is to compute $\boldsymbol{\alpha}$ by setting the negative entries in $\mathbf{K}$ to zero.

For completeness, the dampened Sinkhorn algorithm is included in Algorithm 1. Given n features of d dimensions, computing the kernel matrix takes $\mathcal{O}(n^2d)$, whereas each Sinkhorn iteration takes $\mathcal{O}(n^2)$ time. In practice, 10 iterations are sufficient to find a good solution. The damping factor $\tau = 0.5$ is typically used. This slows the convergence rate but avoids oscillations and other numerical issues associated with the undampened version ($\tau = 1$).

Algorithm 1 Dampened Sinkhorn Algorithm

```
1: procedure SINKHORN($\mathbf{K}, \tau, \mathrm{T}$)
2:     $\boldsymbol{\alpha} \leftarrow \mathbf{1}_n$
3:     for $t = 1$ to T do
4:         $\boldsymbol{\sigma} = \mathtt{diag}(\boldsymbol{\alpha})\mathbf{K}\mathtt{diag}(\boldsymbol{\alpha})\mathbf{1}_n$
5:         $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}/\boldsymbol{\sigma}^{\tau}$
6:     return $\boldsymbol{\alpha}$
```

γ-democratic aggregation. We propose a parametrized family of democratic aggregation functions that interpolate between sum pooling and fully democratic pooling. Given a parameter $0 \leq \gamma \leq 1$, the γ-democratic aggregation is obtained by solving for a vector of weights $\boldsymbol{\alpha}$ such that:

$$\mathtt{diag}(\boldsymbol{\alpha})\mathbf{K}\mathtt{diag}(\boldsymbol{\alpha})\mathbf{1}_n = (\mathbf{K}\mathbf{1}_n)^{\gamma}. \tag{6}$$

When $\gamma = 0$, this corresponds to the democratic aggregation, and when $\gamma = 1$, this corresponds to sum aggregation since $\boldsymbol{\alpha} = \mathbf{1}_n$ satisfies the above equation. The above equation can be solved by modifying the update rule for computing σ in the Sinkhorn iterations to:

$$\sigma = \mathtt{diag}(\boldsymbol{\alpha})\mathbf{K}\mathtt{diag}(\boldsymbol{\alpha})\mathbf{1}_n/(\mathbf{K}\mathbf{1}_n)^{\gamma}, \tag{7}$$

in Algorithm 1, where / denotes element-wise division. Thus, the solution can be equally efficient for any value of γ. Intermediate values of γ allow the contributions $C(\mathbf{x})$ of each feature $\mathbf{x}$ within the set to vary and, in our experiments, we find this can lead to better results than the extremes (*i.e.*, $\gamma = 1$).

Second-order democratic aggregation. In practice, features extracted using deep ConvNets can be high-dimensional. For example, an input image I is passed through layers of a ConvNet to obtain a feature map $\mathbf{\Phi}(I)$ of size $W \times H \times D$. Here $d = D$ corresponds to the number of filters in the convolutional layer and $n = W \times H$ corresponds to the spatial resolution of the feature. For state-of-the-art ConvNets from which features are typically extracted, the values of n and d are comparable and in the range of a few hundred to a thousand. Thus, explicitly realizing the outer products can be expensive. Below we show several properties of democratic aggregation with outer-product encoders. Some of these properties allow aggregation in a computationally and memory efficient manner.

Proposition 1. *For outer-product encoders, the solution to the γ-democratic kernels exists for all values of γ as long as $||\mathbf{x}|| > 0,\ \forall \mathbf{x} \in \mathcal{X}$.*

Proof. For the outer-product encoder we have:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}') = \mathrm{vec}(\mathbf{x}\mathbf{x}^T)^T\,\mathrm{vec}(\mathbf{x}'\mathbf{x}'^T) = (\mathbf{x}^T\mathbf{x}')^2 \geq 0.$$

Thus, all the entries of the kernel matrix are non-negative and the kernel matrix is strictly positive definite when $||\mathbf{x}|| > 0,\ \forall \mathbf{x} \in \mathcal{X}$. This is a sufficient condition for the solution to exist [23]. Note that the kernel matrix of the outer product encoders is positive even when $\mathbf{x}^T\mathbf{x}' < 0$.

Proposition 2. *For outer-product encoders, the solution $\boldsymbol{\alpha}$ to the γ-democratic kernels can be computed in $\mathcal{O}(n^2 d)$ time and $\mathcal{O}(n^2 + nd)$ space.*

Proof. The running time of the Sinkhorn algorithm is dominated by the time to compute the kernel matrix $\mathbf{K}$. Naively computing the kernel matrix for d^2 dimensional features would take $\mathcal{O}(n^2d^2)$ time and $\mathcal{O}(n^2+nd^2)$ space. However, since the kernel entries of the outer products are just the square of the kernel entries of the features before the encoding step, one can compute the kernel $\mathbf{K}$ by simply squaring the kernel of the raw features, which can be computed in $\mathcal{O}(n^2d)$ time and $\mathcal{O}(n^2 + nd)$ space. Thus the weights $\boldsymbol{\alpha}$ for the second-order features can also be computed in $\mathcal{O}(n^2d)$ time and $\mathcal{O}(n^2 + nd)$ space.

Proposition 3. *For outer-product encoders, γ-democratic aggregation $\xi(\mathcal{X})$ can be computed with low-memory overhead using Tensor Sketching.*

Proof. Let θ be a low-dimensional embedding that approximates the inner product between two outer-products, *i.e.*,

$$\theta(\mathbf{x})^T\theta(\mathbf{x}') \sim \mathrm{vec}(\mathbf{x}\mathbf{x}^T)^T\,\mathrm{vec}(\mathbf{x}'\mathbf{x}'^T), \tag{8}$$

and $\theta(\mathbf{x}) \in \mathbb{R}^k$ with $k << d^2$. Since the γ-democratic aggregation of $\mathcal{X}$ is a linear combination of the outer-products, the overall feature $\xi(\mathcal{X})$ can be written as:

$$\xi(\mathcal{X}) = \sum_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x})\mathbf{x}\mathbf{x}^T \sim \sum_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x})\theta(\mathbf{x}). \tag{9}$$

Thus, instead of realizing the overall feature $\xi(\mathcal{X})$ of size d^2, one can use the embedding θ to obtain a feature of size k as a democratic aggregation of the approximate outer-products. One example of an approximate outer-product embedding is the Tensor Sketching (TS) approach of Pham and Pagh [42]. Tensor sketching has been used to approximate second-order sum pooling [12] resulting in an order-of-magnitude savings in space at a marginal loss in performance on classification tasks. Our experiments show that sketching also performs well in the context of democratic aggregation.

3.2 Spectral normalization of second-order representations

A different line of work [6,33,31,58] has investigated matrix functions to normalize the second-order representations obtained by sum pooling. For example, the improved bilinear pooling [33] and second-order approaches [24,25,28] construct a global representation by sum pooling of outer-products:

$$\mathbf{A} = \sum_{\mathbf{x}\in\mathcal{X}} \mathbf{x}\mathbf{x}^T. \tag{10}$$

The matrix $\mathbf{A}$ is subsequently normalized using matrix power function $\mathbf{A}^p$ with $0 < p < 1$. When $p = 1/2$, this corresponds to the matrix square-root which is defined as matrix $\mathbf{Z}$ such that $\mathbf{Z}\mathbf{Z} = \mathbf{A}$. Matrix function can be computed using the Singular Value Decomposition (SVD). Given matrix $\mathbf{A}$ with a SVD given by $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T$, where the matrix $\Lambda = \mathtt{diag}(\lambda_1, \lambda_2, ..., \lambda_d)$, with $\lambda_i \geq \lambda_{i+1}$, the matrix function f can be written as $\mathbf{Z} = f(\mathbf{A}) = \mathbf{U}g(\Lambda)\mathbf{U}^T$, where g is applied to the elements in the diagonal of Λ. Thus, the matrix power can be computed as $\mathbf{A}^p = \mathbf{U}\Lambda^p\mathbf{U}^T = \mathbf{U}\mathtt{diag}(\lambda_1^p, \lambda_2^p, ..., \lambda_d^p)\mathbf{U}^T$. Such spectral normalization techniques scale the spectrum of the matrix $\mathbf{A}$. The following establishes a connection between the spectral normalization techniques and democratic pooling.

Let $\hat{\mathbf{A}}^p$ be the ℓ_2 normalized version of $\mathbf{A}^p$ and $r_{\max}$ and $r_{\min}$ be the maximum and minimum squared radii of the data $\mathbf{x} \in \mathcal{X}$ defined as:

$$r_{\max} = \max_{\mathbf{x}\, \in \mathcal{X}} ||\mathbf{x}||^2, \; r_{\min} = \min_{\mathbf{x}\, \in \mathcal{X}} ||\mathbf{x}||^2. \tag{11}$$

As earlier, let $C(\mathbf{x})$ be the contribution of the vector $\mathbf{x}$ to the the aggregated representation defined as:

$$C(\mathbf{x}) = \mathrm{vec}(\mathbf{x}\mathbf{x}^T)^T \, \mathrm{vec}(\hat{\mathbf{A}}^p). \tag{12}$$

Proposition 4. *The following properties hold true:*

1. *The ℓ_2 norm of $\mathrm{vec}(\mathbf{A}^p)$ is $\rho(\mathbf{A}^p) = ||\,\mathrm{vec}(\mathbf{A}^p)|| = \left(\sum_i \lambda_i^{2p}\right)^{1/2}$.*
2. *$\sum_{\mathbf{x}\in\mathcal{X}} C(\mathbf{x}) = \mathtt{Trace}(\mathbf{A}^{1+p}/||\mathbf{A}^p||) = \left(\sum_i \lambda_i^{1+p}\right)/\rho(\mathbf{A}^p)$.*
3. *The maximum value $M = \max_{\mathbf{x}\in\mathcal{X}} C(\mathbf{x}) \leq r_{\max}\lambda_1^p/\rho(\mathbf{A}^p)$.*
4. *The minimum value $m = \min_{\mathbf{x}\in\mathcal{X}} C(\mathbf{x}) \geq r_{\min}\lambda_d^p/\rho(\mathbf{A}^p)$.*

Proof. The proof is left in the supplementary material.

Proposition 5. *The variance σ^2 of the contributions $C(\mathbf{x})$ satisfies*

$$\sigma^2 \leq (M-\mu)(\mu-m) \leq \frac{(M-m)^2}{4} \leq \frac{r_{\max}^2\lambda_1^{2p}}{4\rho(\mathbf{A}^p)^2}, \tag{13}$$

where M and m are the maximum and minimum values defined above and μ is the mean of $C(\mathbf{x})$ given by $\sum_{\mathbf{x}\in\mathcal{X}} C(\mathbf{x})/n$ where n is the cardinality of $\mathcal{X}$. All of the above quantities can be computed from the spectrum of the matrix $\mathbf{A}$.

Proof. The proof can be obtained by a straightforward application of Popoviciu's inequality on variances [43] and a tighter variant by Bhatia and Davis [4]. The last inequality is obtained by setting $m = 0$.

The above shows that smaller values p reduce an upper-bound on the variance of the contributions thereby equalizing their contributions. The upper bound is a monotonic function of the exponent p and is minimized when $p = 0$ reducing all the spectrum to an identity matrix. This corresponds to whitening of the matrix $\mathbf{A}$. However, complete whitening often leads to poor results while intermediate values such as $p = 1/2$ can be significantly better than $p = 1$ [24,25,33,31]. In the experiments section we evaluate these bounds on deep features from real data.

Proposition 6. *For exponents $0 < p < 1$, the matrix power $\mathbf{A}^p$ may not lie in the linear span of the outer-products of the features $\mathbf{x} \in \mathcal{X}$.*

The proof of Proposition 6 is left in the supplementary material. A consequence of this is that the matrix power cannot be easily computed in the low-dimensional embedding space of outer-products encoding such as Tensor Sketch. It does however lie in the linear span of the outer-products of the eigenvectors. However, computing eigenvectors can be significantly slower than computing weighted aggregates. We describe the computation and memory trade-offs between computing the matrix powers and democratic pooling in Section 4.5.

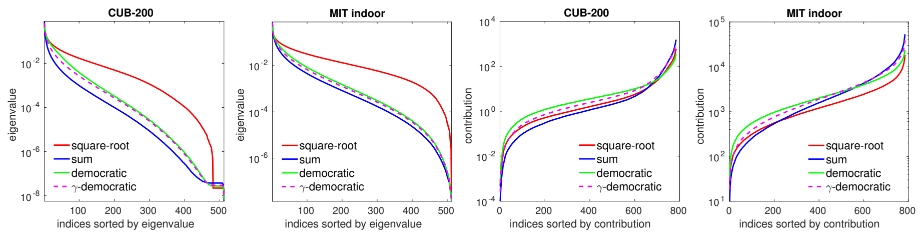
4 Experiments

We analyze the behavior of matrix power normalization and γ-democratic pooling empirically on several fine-grained and texture recognition datasets. The general experiment setting and the datasets are described in Section 4.1. We validate the theoretical bounds on the feature contributions with real data in Section 4.2. We compare our models against sum-pooling baseline, matrix power normalization, and other state-of-the-art methods in Sections 4.3 and 4.4. Finally, we include a discussion on runtime and memory consumption for various approaches and a technique to perform end-to-end fine-tuning in Section 4.5.

4.1 Experimental setup

Datasets. We experiment on Caltech-UCSD Birds [56], Stanford Cars [29] and FGVC Aircrafts [35] datasets. Birds dataset contains 11,788 images which contain over 200 bird species. Stanford Cars dataset consists of 16.185 images across 196 categories and FGVC Aircrafts provides 10,000 images of 100 categories. For each dataset, we use the train and test splits provided by the benchmarks and only the corresponding category labels are used during training phase. In addition to the above fine-grained classification tasks, we also analyze the performance of various approaches on the following datasets: Describable Texture Dataset (DTD) [8], Flickr Material Dataset (FMD) [48] and MIT indoor scene dataset [45]. DTD consists of 5,640 images across 47 texture attributes. We report results averaged over the 10 splits provided by the dataset. FMD provides 1000 images from 10 different material categories. We randomly split half of images for training and the rest for testing for each category and report results across multiple splits. The MIT indoor scene dataset contains 67 indoor scene categories, each of which includes 80 images for training and 20 for testing.

Features. We aggregate the second-order features with γ-democratic pooling and matrix power normalization using VGG-16 [50] and ResNet101 [16] networks. We follow the work [34] and resize input images to 448×448 and aggregate the last convolutional layer features after ReLU activations. For the VGG-16 network architecture, this results in feature maps of size $28 \times 28 \times 512$



(a) spectrum (eigenvalues) (b) contributions $C(\mathbf{x})$

Fig. 1. (a) The spectrum (eigenvalues) for various feature aggregators on CUB-200 and MIT indoor datasets. (b) The individual feature vector contributions $C(\mathbf{x})$.

(before aggregation), while for the ResNet101 architecture this results in maps of size $14 \times 14 \times 2048$. For γ-democratic pooling, we run the modified Sinkhorn algorithm for 10 iterations with the power exponent $\tau = 0.5$. Fully democratic pooling [37] and sum pooling can be implemented by setting $\gamma = 0$ and $\gamma = 1$, respectively. The aggregated features are followed by element-wise signed square-root and ℓ_2 normalization. For fine-grained recognition datasets, we aggregate the VGG-16 features fine-tuned with vanilla BCNN models, while the ImageNet pretrained networks without fine-tuning are used for texture and scene datasets.

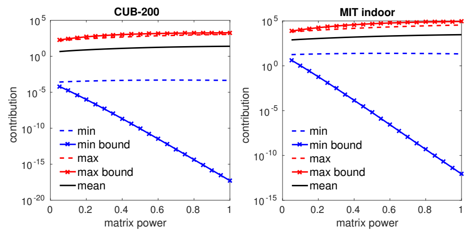
4.2 The distribution of the spectrum and feature contributions

In this section, we analyze how democratic pooling and matrix normalization effect the spectrum (set of eigenvalues) of the aggregated representation, as well as how the contributions of individual features are distributed as a function of γ for the democratic pooling and p of the matrix power normalization.

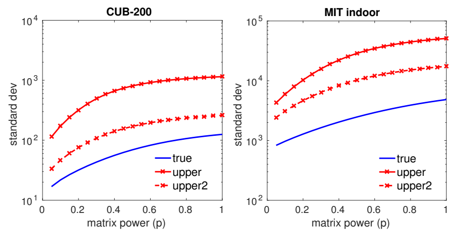
We randomly sampled 50 images from CUB and MIT indoor datasets each and plotted the spectrum (normalized to unit length) and the feature vector contributions $C(\mathbf{x})$ (Eq. (12)) in Figure 1. In this experiment, we use the matrix power $p = 0.5$ and $\gamma = 0.5$. Figure 1(a) shows that the square root yields a flatter spectrum in comparison to the sum aggregation. Democratic aggregation distributes the energy away from the top eigenvalues but has considerably sharper spectrum in comparison to the square root. The γ-democratic pooling interpolates between sum and fully democratic pooling.

Figure 1(b) shows the contributions of each feature $\mathbf{x}$ to the aggregate for different pooling techniques (Eq. (12)). The contributions are more evenly distributed for the matrix square root in comparison to sum pooling. Democratic pooling flattens the individual contributions the most – we note that it is explicitly designed to have this effect. These two plots show that democratic aggregation and power normalization both achieve equalization of feature contributions.

Figure 2 shows the variances of the contributions $C(\mathbf{x})$ to the aggregation $\hat{\mathbf{A}}^p$ using the VGG-16 features for different values of the exponent p. Figure 2(a) shows the true minimum, maximum, mean as well as the bounds of these quanti-



(a) bounds on contribution



(b) bounds on variance

Fig. 2. (a) The upper (red solid) and lower bounds (blue solid) on the contributions to the set similarity versus the exponent of matrix power normalization on Birds and MIT indoor datasets. Maximum and minimum values are shown in dashed lines and the the mean is shown in black solid lines. (b) The upper bounds to the variance of feature contributions $C(\mathbf{x})$.

ties expressed in Proposition 4. The upper bound on the maximum contribution, *i.e.*, $r_{\max}\lambda_1^p/\rho(\mathbf{A}^p)$, is tight on both datasets, as can be seen in the overlapping red lines, while the lower bound is significantly less tight.

Figure 2(b) shows the true deviation and two different upper bounds on the variance of the contributions as expressed in Proposition 5 and Eq. (13). The tighter bound shown by the dashed red line corresponds to the version with the mean μ in Eq. (13). The plot shows that the matrix power normalization implicitly reduces the variance in feature contributions similar to equalizing the feature vector contributions $C(\mathbf{x})$ in democratic aggregation. These plots are averaged over 50 examples from the CUB-200 and MIT indoor datasets.

4.3 Effect of γ on democratic pooling

Table 1 shows the performance as a function of γ for the γ-democratic pooling and p for the matrix normalization on the VGG-16 network. For DTD dataset, we report results on the first split. For FMD dataset, we randomly sample half of the data in each category for training and use the rest for testing. We use the standard training and testing splits on remaining datasets. We augment the training set by flipping its images and train k one-vs-all linear SVM classifiers with hyperparameter $C = 1$. At the test time, we average predictions from an image and its flipped copy. Optimal γ and the matrix power p are also reported.

The results on sum pooling correspond to the symmetric BCNN models [33]. Fully democratic pooling (γ=0) improves the performance over sum pooling by 0.7-1%. However, equalizing feature contributions hurts performance on Stanford Cars and FMD dataset. Table 1 shows that reducing the contributions by adjusting $0 < \gamma < 1$ helps outperform sum pooling and fully democratic pooling.

Matrix power normalization outperforms γ-democratic pooling by 0.2-1%. However, computing the matrix powers on covariance matrices is computationally expensive compared to our democratic aggregation. We discuss these trade-offs in the Section 4.5.

Dataset	γ-democratic			$\mathbf{A}^p$
	Democratic γ=0	Optimal γ	Sum $\gamma = 1$	
Caltech UCSD Birds	84.7	84.9 (0.5)	84.0	85.9 (0.3)
Stanford Cars	89.7	90.8 (0.5)	90.6	91.7 (0.5)
FGVC Aircrafts	86.7	86.7 (0.0)	85.7	87.6 (0.3)
DTD	72.2	72.3 (0.3)	71.2	72.9 (0.6)
FMD	82.8	84.8 (0.8)	84.6	85.0 (0.7)
MIT indoor	79.6	80.4 (0.3)	79.5	80.9 (0.6)

Table 1. The accuracy of aggregating second-order features w.r.t. various aggregators using fine-tuned VGG-16 on fine-grained recognition (top) and using ImageNet pretrained VGG-16 on other (bottom) datasets. From left to right, we vary γ values and compare democratic pooling, γ-democratic pooling and average pooling with the matrix power aggregation. The optimal values of γ and p are indicated in parentheses.

4.4 Democratic pooling with Tensor Sketching

One of the main advantages of the democratic pooling approaches over matrix power normalization techniques is that the embeddings can be computed in a low-dimensional space using tensor sketching. To demonstrate this advantage, we compute the second-order democratic pooling combined with tensor sketching on 2048 dimensional ResNet-101 features. Direct construction of second-order features yields ∼4M dimensional features which are impractical to manipulate on GPU/CPU. Therefore, we apply the Tensor Sketch [42] to approximate the outer product using 8192 dimensional features, which is far lower than 2048^2 of the full outer product. The features are aggregated using γ-democratic approach with $\gamma = 0.5$. We compare our method to the state of the art on MIT indoor, FMD and DTD datasets. We report the mean accuracy. For DTD and FMD, we also indicate the standard deviation over 10 splits.

Results on MIT indoor. Table 2 reports the accuracy on MIT indoor. The baseline model approximating second-order features with tensor sketch followed by sum pooling achieves 82.8% accuracy. With democratic pooling, our model achieves state-of-the-art accuracy of 84.3% which is 1.5% more than the baseline. Moreover, Table 1 shows that we outperform the matrix power normalization using VGG-16 network by 3.4%. Note that (i) matrix power normalization is impractical for ResNet101 features, (ii) it cannot be computed by sketching due to Proposition 6. We also outperform FASON [10] by 2.6%. FASON fuses the first- and second-order features from *conv*4_4 and *conv*5_4 layers of the VGG-19 networks given 448×448 image size and scores 81.7% accuracy. Recent work on Spectral Features [22] achieves the same accuracy as our best model with democratic pooling. However, approach [22] uses more data augmentations (rotation, shifts, *etc.*) during training and pretrains the VGG-19 network on the large-scale Places205 dataset. In contrast, our networks are pretrained on ImageNet which arguably has a larger domain shift from the MIT indoor dataset than Places205.

Method		accuracy
Places-205	[54]	80.9
Deep Filter Banks	[9]	81.0
Spectral Features	[22]	84.3
FASON	[10]	81.7
ResNet101 + TS + sum pooling	*(baseline)*	82.8
ResNet101 + TS + γ-democratic	*(ours)*	**84.3**

Table 2. Evaluations and comparisons to the state of the art on MIT indoor dataset.

Results on FMD. Table 3 compares the accuracy on FMD dataset. Recent work on Deep filter banks [9], denoted as FV+FC+CNN, which combines fully-connected CNN features and Fisher Vector approach, scores 82.1% accuracy. In contrast to several methods, FASON uses single-scale input images (224×224) and also scores 82.1% accuracy. Our second-order democratic pooling outperforms FASON by 0.7% given the same image size. For 448×448 image size, our model scores 84.3% and outperforms other state-of-the-art approaches.

Method		input size	accuracy
IFV+DeCAF	[8]	ms	65.5 ± 1.3
FV+FC+CNN	[9]	ms	82.2 ± 1.4
LFV	[51]	ms	82.1 ± 1.9
SMO Task	[60]	-	82.3 ± 1.7
FASON	[10]	224	82.1 ± 1.9
ResNet101 + TS + sum pooling	*(baseline)*	448	83.7 ± 1.3
ResNet101 + TS + γ-democratic	*(ours)*	448	**84.3** ± 1.5
ResNet101 + TS + γ-democratic	*(ours)*	224	82.8 ± 2.5

Table 3. Evaluations and comparisons to the state of the art on the FMD dataset. The middle column indicates the image size used by each method (*ms* indicates multiple scales while hyphen denotes an unknown size).

Results on DTD. Table 4 presents our results and comparisons on DTD dataset. Deep filter banks [9], denoted as FV+FC+CNN, reports 75.5% accuracy. Combined second-order features and tensor sketching outperforms Deep filter banks by 0.3%. With second-order democratic pooling and 448×448 size images, our model achieves 76.2% accuracy and outperforms FV+FC+CNN 0.7%. Note that FV+FC+CNN exploits several scales of image sizes.

4.5 Discussion

While matrix power normalization achieves marginally better performance, it requires SVD which is computationally expensive and not GPU friendly *e.g.*, the CUDA BLAS cannot perform SVD for large matrices. Even in the case of

Method		input size	accuracy
LFV	[51]	ms	73.8 ± 1.0
FV+FC+CNN	[9]	ms	75.5 ± 0.8
FASON	[10]	224	72.9 ± 0.7
ResNet101 + TS + sum pooling	*(baseline)*	448	75.8 ± 0.7
ResNet101 + TS + γ-democratic	*(ours)*	448	**76.2** ± 0.7
ResNet101 + TS + γ-democratic	*(ours)*	224	73.0 ± 0.6

Table 4. Evaluations and comparisons to the state of the art on the DTD dataset. The middle column indicates the image size used by each method (*ms* indicates multiple scales while hyphen denotes an unknown size).

matrix square root which can be approximated via Newton's iterations [33], the iterations involve matrix-matrix multiplication of $\mathcal{O}(n^3)$ complexity. In contrast, solving democratic pooling via the Sinkhorn algorithm (Algorithm 1) involves only matrix-vector multiplication which is $\mathcal{O}(n^2)$. Empirically, we find that solving Sinkhorn iterations is an order of magnitude faster than solving the matrix square root on a NVIDIA Titan X GPU. Moreover, the complexity of Sinkhorn iteration depends only on the kernel matrix – it is independent of the feature vector size. In contrast, the memory required by a covariance matrix grows with $\mathcal{O}(n^2)$ which becomes prohibitive for feature vectors greater than 512 dimensions. Second-order democratic pooling with tensor sketching yields comparable results and reduces the memory usage by two orders of magnitude over the matrix power normalization.

Although we did not report results using end-to-end training, one can easily obtain the gradients of the Sinkhorn algorithm using automatic differentiation by implementing Algorithm 1 in a library such as PyTorch or Tensorflow. Training using gradients from iterative solvers has been performed in a number of applications (*e.g.*, [13] and [36]) which suggests that it is a promising direction.

5 Conclusions

We proposed a second-order aggregation method referred to as γ-democratic pooling that interpolates between sum (γ=1) and democratic pooling (γ=0) and outperforms other aggregation approaches on several classification tasks. We demonstrated that our approach enjoys low computational complexity compared to the matrix square root approximations via Newton's iterations. With the use of sketching, our approach is not limited to aggregating small feature vectors which is typically the case for the matrix power normalization. The source code for the project is available at http://vis-www.cs.umass.edu/o2dp.

Acknowlegements. We acknowledge support from NSF (#1617917, #1749833) and the MassTech Collaborative grant for funding the UMass GPU cluster.

References

1. Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: CVPR (2016)
2. Arsigny, V., Fillard, P., Pennec, X., Ayache, N.: Log-euclidean metrics for fast and simple calculus on diffusion tensors. Magnetic resonance in medicine **56**(2), 411–421 (2006)
3. Bhatia, R.: Positive definite matrices. Princeton Univ Press (2007)
4. Bhatia, R., Davis, C.: A better bound on the variance. The American Mathematical Monthly **107**(4), 353–357 (2000)
5. Boughorbel, S., Tarel, J.P., Boujemaa, N.: Generalized Histogram Intersection Kernel for Image Recognition. In: ICIP (2005)
6. Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C.: Semantic Segmentation with Second-Order Pooling. In: ECCV (2012)
7. Cherian, A., Sra, S., Banerjee, A., Papanikolopoulos, N.: Jensen-Bregman LogDet Divergence with Application to Efficient Similarity Search for Covariance Matrices. TPAMI **35**(9), 2161–2174 (2013)
8. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing Textures in the Wild. In: CVPR (2014)
9. Cimpoi, M., Maji, S., Vedaldi, A.: Deep Filter Banks for Texture Recognition and Segmentation. In: CVPR (2015)
10. Dai, X., Yue-Hei Ng, J., Davis, L.S.: FASON: First and Second Order Information Fusion Network for Texture Recognition. In: CVPR (2017)
11. Dryden, I.L., Koloydenko, A., Zhou, D.: Non-euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. The Annals of Applied Statistics **3**(3), 1102–1123 (2009)
12. Gao, Y., Beijbom, O., Zhang, N., Darrell, T.: Compact bilinear pooling. In: CVPR (2016)
13. Genevay, A., Peyré, G., Cuturi, M.: Learning generative models with sinkhorn divergences. arXiv preprint arXiv:1706.00292 (2017)
14. Gong, Y., Wang, L., Guo, R., Lazebnik, S.: Multi-scale Orderless Pooling of Deep Convolutional Activation Features. In: ECCV (2014)
15. Guo, K., Ishwar, P., Konrad, J.: Action recognition from video using feature covariance matrices. Trans. Img. Proc. **22**(6), 2479–2494 (2013)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: CVPR (2016)
17. Huang, Z., Gool, L.V.: A Riemannian Network for SPD Matrix Learning. In: AAAI (2017)
18. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix Backpropagation for Deep Networks with Structured Layers. In: ICCV (2015)
19. Jégou, H., Douze, M., Schmid, C.: On the Burstiness of Visual Elements. In: CVPR (2009)
20. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating Local Descriptors into a Compact Image Representation. In: CVPR (2010)
21. Jégou, H., Zisserman, A.: Triangulation embedding and democratic aggregation for image search. In: CVPR (2014)
22. Khan, S.H., Hayat, M., Porikli, F.: Scene Categorization with Spectral Features. In: ICCV (2017)
23. Knight, P.A.: The sinkhorn-knopp algorithm: Convergence and applications. SIAM J. Matrix Anal. Appl. **30**(1), 261–275 (Mar 2008)

24. Koniusz, P., Yan, F., Gosselin, P., Mikolajczyk, K.: Higher-order Occurrence Pooling on Mid- and Low-level Features: Visual Concept Detection. Technical Report, HAL Id: hal-00922524 (2013)
25. Koniusz, P., Yan, F., Gosselin, P., Mikolajczyk, K.: Higher-order occurrence pooling for bags-of-words: Visual concept detection. PAMI **39**(2), 313–326 (2017)
26. Koniusz, P., Cherian, A., Porikli, F.: Tensor representations via kernel linearization for action recognition from 3d skeletons. In: ECCV. pp. 37–53. Springer (2016)
27. Koniusz, P., Tas, Y., Zhang, H., Harandi, M., Porikli, F., Zhang, R.: Museum exhibit identification challenge for the supervised domain adaptation. In: ECCV (2018)
28. Koniusz, P., Zhang, H., Porikli, F.: A deeper look at power normalizations. In: CVPR. pp. 5774–5783 (2018)
29. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D Object Representations for Fine-Grained Categorization. In: Workshop on 3D Representation and Recognition (3dRR) (2013)
30. Li, P., Wang, Q.: Local Log-Euclidean Covariance Matrix (L^2 ECM) for Image Representation and Its Applications. In: ECCV (2012)
31. Li, P., Xie, J., Wang, Q., Zuo, W.: Is Second-order Information Helpful for Large-scale Visual Recognition? In: ICCV (2017)
32. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear Convolutional Neural Networks for Fine-Grained Visual Recognition. IEEE TPAMI **40**(6) (2018)
33. Lin, T.Y., Maji, S.: Improved Bilinear Pooling with CNNs. In: BMVC (2017)
34. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear CNN Models for Fine-grained Visual Recognition. In: ICCV (2015)
35. Maji, S., Kannala, J., Rahtu, E., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft (2013)
36. Mena, G., Belanger, D., Linderman, S., Snoek, J.: Learning latent permutations with gumbel-sinkhorn networks. arXiv preprint arXiv:1802.08665 (2018)
37. Murray, N., Jégou, H., Perronnin, F., Zisserman, A.: Interferences in match kernels. IEEE TPAMI **39**(9), 1797–1810 (2017)
38. Negrel, R., Picard, D., Gosselin, P.H.: Compact Tensor Based Image Representation for Similarity Search. ICIP (2012)
39. Pennec, X., Fillard, P., Ayache, N.: A Riemannian Framework for Tensor Computing. IJCV **66**(1), 41–66 (2006)
40. Perronnin, F., Dance, C.: Fisher Kernels on Visual Vocabularies for Image Categorization. In: CVPR (2007)
41. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher Kernel for Large-Scale Image Classification. In: ECCV (2010)
42. Pham, N., Pagh, R.: Fast and scalable polynomial kernels via explicit feature maps. In: KDD (2013)
43. Popoviciu, T.: Sur les équations algébriques ayant toutes leurs racines réelles. Mathematica **9**, 129–145 (1935)
44. Porikli, F., Tuzel, O.: Covariance Tracker. In: CVPR (2006)
45. Quattoni, A., Torralba, A.: Recognizing Indoor Scenes. In: CVPR (2009)
46. Romero, A., Terán, M.Y., Gouiffès, M., Lacassagne, L.: Enhanced local binary covariance matrices for texture analysis and object tracking. MIRAGE (2013)
47. Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: Theory and practice. IJCV **105**(3), 222–245 (2013)
48. Sharan, L., Rosenholtz, R., H., A.E.: Material perceprion: What can you see in a brief glance? Journal of Vision **9:784**(8) (2009)

49. Shih, Y.F., Yeh, Y.M., Lin, Y.Y., Weng, M.F., Lu, Y.C., Chuang, Y.Y.: Deep Co-occurrence Feature Learning for Visual Object Recognition. In: CVPR (2017)
50. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
51. Song, Y., Zhang, F., Li, Q., Huang, H., O'Donnell, L.J., Cai, W.: Locally-transferred fisher vectors for texture classification. In: ICCV (Oct 2017)
52. Tuzel, O., Porikli, F., Meer, P.: Region Covariance: A Fast Descriptor for Detection and Classification. In: ECCV (2006)
53. Tuzel, O., Porikli, F., Meer, P.: Pedestrian Detection via Classification on Riemannian Manifolds. IEEE TPAMI **30**(10), 1713–1727 (2008)
54. Wang, L., Guo, S., Huang, W., Qiao, Y.: Places205-VGGnet models for scene recognition. CoRR **abs/1508.01667** (2015)
55. Wang, Z., Vemuri, B.C.: An affine invariant tensor dissimilarity measure and its applications to tensor-valued image segmentation. In: CVPR (2004)
56. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-UCSD Birds 200. Tech. Rep. CNS-TR-2010-001, California Institute of Technology (2010)
57. Yandex, A.B., Lempitsky, V.: Aggregating Local Deep Features for Image Retrieval. In: ICCV (2015)
58. Yu, K., Salzmann, M.: Second-order Convolutional Neural Networks. abs/1703.06817 (2017)
59. Yu, K., Salzmann, M.: Statistically-motivated second-order pooling. In: ECCV (2018)
60. Zhang, Y., Ozay, M., Liu, X., Okatani, T.: Integrating deep features for material recognition. In: ICPR (2016)