Low-rank Bilinear Pooling for Fine-Grained Classification

Shu Kong, Charless Fowlkes
Dept. of Computer Science, University of California, Irvine
{skong2, fowlkes}@ics.uci.edu

Abstract

Pooling second-order local feature statistics to form a high-dimensional bilinear feature has been shown to achieve state-of-the-art performance on a variety of fine-grained classification tasks. To address the computational demands of high feature dimensionality, we propose to represent the covariance features as a matrix and apply a low-rank bilinear classifier. The resulting classifier can be evaluated without explicitly computing the bilinear feature map which allows for a large reduction in the compute time as well as decreasing the effective number of parameters to be learned.

To further compress the model, we propose a classifier co-decomposition that factorizes the collection of bilinear classifiers into a common factor and compact per-class terms. The co-decomposition idea can be deployed through two convolutional layers and trained in an end-to-end architecture. We suggest a simple yet effective initialization that avoids explicitly first training and factorizing the larger bilinear classifiers. Through extensive experiments, we show that our model achieves state-of-the-art performance on several public datasets for fine-grained classification trained with only category labels. Importantly, our final model is an order of magnitude smaller than the recently proposed compact bilinear model [8], and three orders smaller than the standard bilinear CNN model [19].

1. Introduction and Related Work

Fine-grained categorization aims to distinguish subordinate categories within an entry-level category, such as identifying the bird species or particular models of aircraft. Compared to general purpose visual categorization problems, fine-grained recognition focuses on the characteristic challenge of making subtle distinctions (low inter-class variance) despite highly variable appearance due to factors such as deformable object pose (high intra-class variance). Fine-grained categorization is often made even more challenging by factors such as large number of categories and the lack of training data.

One approach to dealing with such nuisance parameters has been to exploit strong supervision, such as detailed part-level, keypoint-level and attribute annotations [37, 9, 35]. These methods learn to localize semantic parts or keypoints and extract corresponding features which are used as a holistic representation for final classification. Strong supervision with part annotations has been shown to significantly improve the fine-grained recognition accuracy. However, such supervised annotations are costly to obtain.

To alleviate the costly collection of part annotations, some have proposed to utilize interactive learning [6]. Partially supervised discovery of discriminative parts from category labels is also a compelling approach, especially given the effectiveness of training with web-scale datasets [16]. One approach to unsupervised part discovery [27, 26] uses saliency maps, leveraging the observation that sparse deep CNN feature activations often correspond to semantically meaningful regions [34, 20]. Another recent approach [32] selects parts from a pool of patch candidates by searching over patch triplets, but relies heavily on training images being aligned w.r.t the object pose. Spatial transformer networks [10] are a very general formulation that explicitly model latent transformations that align feature maps prior to classification. They can be trained end-to-end using only classification loss and have achieved state-of-the-art performance on the very challenging CUB bird dataset [31], but the resulting models are large and stable optimization is non-trivial.

Recently, a surprisingly simple method called bilinear pooling [19] has achieved state-of-the-art performance on a variety of fine-grained classification problems. Bilinear pooling collects second-order statistics of local features over a whole image to form a holistic representation for classification. Second-order or higher-order statistics have been explored in a number of vision tasks (see e.g. [2, 14]). In the context of fine-grained recognition, spatial pooling introduces invariance to deformations while second-order statistics maintain selectivity.

However, the representational power of bilinear features comes at the cost of very high-dimensional feature representations (see Figure 1 (b)), which induce substantial

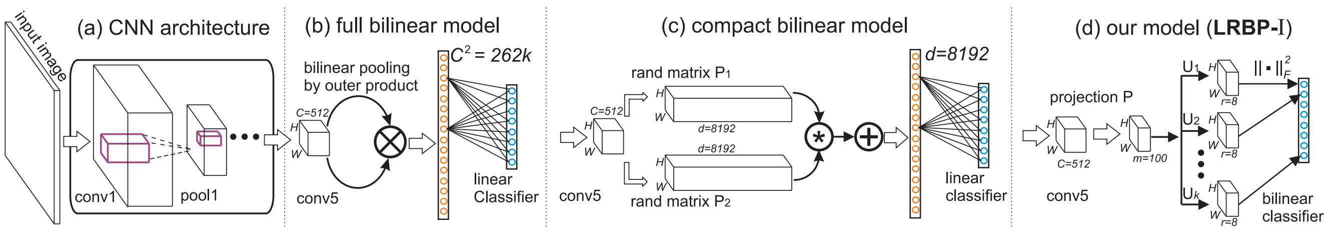

Figure 1: We explore models that perform classification using second order statistics of a convolutional feature map (a) as input (e.g., VGG16 layer $conv5\_3$). Architecture of (b) full bilinear model [19], (c) recently proposed compact bilinear model [8], and (d) our proposed low-rank bilinear pooling model (LRBP). Our model captures second order statistics without explicitly computing the pooled bilinear feature, instead using a bilinear classifier that uses the Frobenius norm as the classification score. A variant of our architecture that exploits co-decomposition and computes low-dimensional bilinear features is sketched in Figure 4.

computational burdens and require large quantities of training data to fit. To reduce the model size, Gao *et al.* [8] proposed using compact models based on either random Maclaurin [12] or tensor sketch [23]. These methods approximate the classifier applied to bilinear pooled feature by the Hadamard product of projected local features with a large random matrix (Figure 1 (c)). These compact models maintain similar performance to the full bilinear feature with a 90% reduction in the number of learned parameters.

The original bilinear pooling work of Lin *et al.* and the compact models of Gao *et al.* ignore the algebraic structure of the bilinear feature map; instead they simply vectorize and apply a linear classifier. Inspired by work on the bilinear SVM [24, 33, 13], we instead propose to use a bilinear classifier applied to the bilinear feature which is more naturally represented as a (covariance) matrix. This representation not only preserves the structural information, but also enables us to impose low-rank constraint to reduce the degrees of freedom in the parameter vector to be learned.

Our model uses a symmetric bilinear form, so computing the confidence score of our bilinear classifier amounts to evaluating the squared Frobenius norm of the projected local features. We thus term our mechanism maximum Frobenius margin. This means that, at testing time, we do not need to explicitly compute the bilinear features, and thus computational time can be greatly reduced under some circumstances, *e.g.*, when the channel number is larger than spatial size. We show empirically this results in improved classification performance, reduces the model size and accelerates feed-forward computation at test time.

To further compress the model for multi-way classification tasks, we propose a simple co-decomposition approach to factorize the joint collection of classifier parameters to obtain a even more compact representation. This multilinear co-decomposition can be implemented using two separate linear convolutional layers, as shown in Figure 1 (d). Rather than first training a set of classifiers and then performing co-decomposition of the parameters, we suggest a simple yet effective initialization based on feature map activation statistics which allows for direct end-to-end training.

We show that our final model achieves the state-of-the-art performance on several public datasets for fine-grained classification by using only the category label. It is worth noting that the set of parameters learned in our model is ten times smaller than the recently proposed compact bilinear model [8], and a hundred times smaller than the original full bilinear CNN model [19].

2. Bilinear Features Meet Bilinear SVMs

To compute the bilinear pooled features for an image, we first feed the image into a convolutional neural network (CNN), as shown in Figure 1 (a), and extract feature maps at a specific layer, say VGG16 $conv5\_3$ after rectification. We denote the feature map by $\mathcal{X} \in \mathbb{R}^{h \times w \times c}$, where h, w and c indicate the height, width and number of feature channels and denote the feature vector at a specific location by $\mathbf{x}_i \in \mathbb{R}^c$ where the spatial coordinate index $i \in [1, hw]$. For each local feature we compute the outer product, $\mathbf{x}_i\mathbf{x}_i^T$ and sum (pool) the resulting matrices over all hw spatial locations to produce a holistic representation of the image of dimension c^2. This computation can be written in matrix notation as $\mathbf{X}\mathbf{X}^T = \sum_{i=1}^{hw} \mathbf{x}_i\mathbf{x}_i^T$, where $\mathbf{X} \in \mathbb{R}^{c \times hw}$ is a matrix by reshaping $\mathcal{X}$ in terms of the third mode. $\mathbf{X}\mathbf{X}^T$ captures the second-order statistics of the feature activations and is closely related to the sample covariance matrix.

In the bilinear CNN model [19] as depicted in Figure 1 (b), the bilinear pooled feature is reshaped into a vector $\mathbf{z} = vec(\mathbf{X}\mathbf{X}^T) \in \mathbb{R}^{c^2}$ and then fed into a linear classifier[1].

Given N training images, we can learn a linear classifier for a specific class parameterized by $\mathbf{w} \in \mathbb{R}^{c^2}$ and bias b. Denote the bilinear feature for image-i by $\mathbf{z}_i$ and its binary class label as $y_i = \pm 1$ for $i = 1, \dots, N$. The standard

[1]Various normalization can be applied here, *e.g.* sign square root power normalization and ℓ_2 normalization. We ignore for now the normalization notations for presentational brevity, and discuss normalization in Section 5.1.

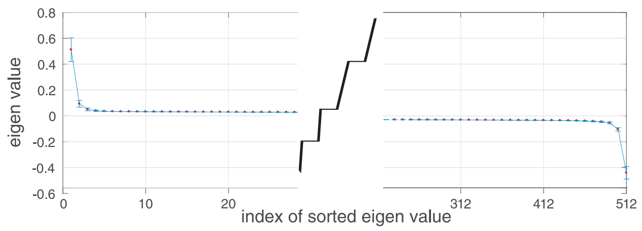

Figure 2: The mean and standard deviation of the eigenvalues the weight matrix $\mathbf{W}$ for 200 linear SVM classifiers applied to bilinear features. As the plot suggests, a large part of the spectrum is typically concentrated around 0 with a few large positive and negative eigenvalues. The middle of the spectrum is excluded here for clarity.

soft-margin SVM training objective is given by:

$$\min_{\mathbf{w},b} \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i \mathbf{w}^T \mathbf{z}_i + b) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \quad (1)$$

2.1. Maximum Frobenius Margin Classifier

We can write an equivalent objective to Equation 1 using the matrix representation of the bilinear feature as:

$$\min_{\mathbf{W},b} \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i \text{tr}(\mathbf{W}^T \mathbf{X}_i \mathbf{X}_i^T) + b) + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 \quad (2)$$

It is straightforward to show that Equation 2 is a convex optimization problem w.r.t. the parameter $\mathbf{W} \in \mathbb{R}^{c\times c}$ and is equivalent to the linear SVM.

Proposition 1 *Let $\mathbf{w}^* \in \mathbb{R}^{c^2}$ be the optimal solution of the linear SVM in Equation 1 over bilinear features, then $\mathbf{W}^* = mat(\mathbf{w}^*) \in \mathbb{R}^{c\times c}$ is the optimal solution in Equation 2. Moreover, $\mathbf{W}^* = \mathbf{W}^{*T}$.*

To give some intuition about this claim, we write the optimal solution to the two SVM problems in terms of the Lagrangian dual variables α associated with each training example:

$$\begin{aligned} \mathbf{w}^* &= \sum_{y_i=1} \alpha_i \mathbf{z}_i - \sum_{y_i=-1} \alpha_i \mathbf{z}_i \\ \mathbf{W}^* &= \sum_{y_i=1} \alpha_i \mathbf{X}_i\mathbf{X}_i^T - \sum_{y_i=-1} \alpha_i \mathbf{X}_i\mathbf{X}_i^T \quad (3) \\ &\text{where} \quad \alpha_i \geq 0, \forall i = 1, \ldots, N, \end{aligned}$$

As $\mathbf{z} = vec(\mathbf{X}\mathbf{X}^T)$, we have $\mathbf{w}^* = vec(\mathbf{W}^*)$ [2]; since $\mathbf{W}^*$ is a sum of symmetric matrices, it must also be symmetric. This expansion motivates us to treat $\mathbf{W}^*$ as the difference of two positive semidefinite matrices corresponding to the

[2]We use $mat(\cdot)$ to denote the inverse of $vec(\cdot)$ so that $vec(mat(\mathbf{w})) = \mathbf{w}$.

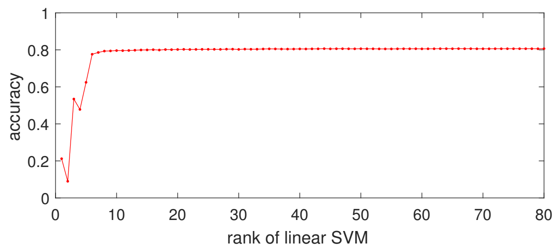

Figure 3: Average accuracy of low-rank linear SVMs. In this experiment we simply use singular value decomposition applied to the set of full rank SVM's for all classes to generate low-rank classifiers satisfying a hard rank constraint (no fine-tuning). Very low rank classifiers still achieve good performance.

positive and negative training examples. It is informative to compare Equation 3 with eigen-decomposition of $\mathbf{W}^*$

$$\begin{aligned} \mathbf{W}^* &= \mathbf{\Psi}\mathbf{\Sigma}\mathbf{\Psi}^T = \mathbf{\Psi}_+\mathbf{\Sigma}_+\mathbf{\Psi}_+^T + \mathbf{\Psi}_-\mathbf{\Sigma}_-\mathbf{\Psi}_-^T \\ &= \mathbf{\Psi}_+\mathbf{\Sigma}_+\mathbf{\Psi}_+^T - \mathbf{\Psi}_-|\mathbf{\Sigma}_-|\mathbf{\Psi}_-^T \quad (4) \\ &= \mathbf{U}_+\mathbf{U}_+^T - \mathbf{U}_-\mathbf{U}_-^T \end{aligned}$$

where $\mathbf{\Sigma}_+$ and $\mathbf{\Sigma}_-$ are diagonal matrices containing only positive and negative eigenvalues, respectively, and $\mathbf{\Psi}_+$ and $\mathbf{\Psi}_-$ are the eigenvectors corresponding to those eigenvalues. Setting $\mathbf{U}_+ = \mathbf{\Psi}_+\mathbf{\Sigma}_+^{\frac{1}{2}}$ and $\mathbf{U}_- = \mathbf{\Psi}_-|\mathbf{\Sigma}_-|^{\frac{1}{2}}$, we have $\mathbf{W} = \mathbf{U}_+\mathbf{U}_+^T - \mathbf{U}_-\mathbf{U}_-^T$.

In general it will *not* be the case that the positive and negative components of the eigendecomposition correspond to the dual decomposition (e.g., that $\mathbf{U}_+\mathbf{U}_+^T = \sum_{y_i=1} \alpha_i \mathbf{X}_i\mathbf{X}_i^T$) since there are many possible decompositions into a difference of psd matrices. However, this decomposition motivates the idea that $\mathbf{W}^*$ may well have a good low-rank decomposition. In particular we know that $rank(\mathbf{W}^*) < min(N, c)$ so if the amount of training data is small relative to c, $\mathbf{W}^*$ will necessarily be low rank. Even with large amounts of training data, SVMs often produce dual variables α which are sparse so we might expect that the number of non-zero entries in $\boldsymbol{\alpha}$ are less than c.

Low rank parameterization: To demonstrate this low-rank hypothesis empirically, we plot in Figure 2 the sorted average eigenvalues with standard deviation of the 200 classifiers trained on bilinear pooled features from the CUB Bird dataset [31]. From the figure, we can easily observe that a majority of eigenvalues are close to zero and an order smaller in magnitude than the largest ones.

This motivates us to impose low-rank constraint to reduce the degrees of freedom in the parameters of the classifier. We use singular value decomposition to generate a low rank approximation of each of the 200 classifiers, discarding those eigenvectors whose corresponding eigenvalue has small magnitude. As shown in Figure 3, a rank 10 approximation of the learned classifier achieves nearly the same classification accuracy as the full rank model. This suggests the set of classifiers can be represented by $512 \times 10 \times 200$ parameters rather than the full set of $512^2 \times 200$ parameters.

Low-rank Hinge Loss: In this paper, we directly impose a hard low-rank constraint $rank(\mathbf{W}) = r \ll c$ by using the parameterization in terms of $\mathbf{U}_+$ and $\mathbf{U}_-$, where $\mathbf{U}_+ \in \mathbb{R}^{c \times r/2}$ and $\mathbf{U}_- \in \mathbb{R}^{c \times r/2}$. This yields the following (non-convex) learning objective:

$$\min_{\mathbf{U}_+, \mathbf{U}_-, b} \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) + \frac{\lambda}{2} R(\mathbf{U}_+, \mathbf{U}_-) \tag{5}$$

where $H(\cdot)$ is the hinge loss and $R(\cdot)$ is the regularizer. The hinge loss can be written as:

$$H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) \equiv \max(0, 1 - y_i\{\mathsf{tr}(\tilde{\mathbf{W}}^T \tilde{\mathbf{X}})\} + b) \tag{6}$$

where

$$\tilde{\mathbf{W}} = \begin{bmatrix} \mathbf{U}_+\mathbf{U}_+^T & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_-\mathbf{U}_-^T \end{bmatrix}, \tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_i\mathbf{X}_i^T & \mathbf{0} \\ \mathbf{0} & -\mathbf{X}_i\mathbf{X}_i^T \end{bmatrix}. \tag{7}$$

While the hinge loss is convex in $\tilde{\mathbf{W}}$, it is no longer convex in the parameters $\mathbf{U}_+, \mathbf{U}_-$ we are optimizing.[3]

Alternately, we can write the score of the low-rank bilinear classifier as a difference of matrix norms which yields the following expression of the hinge-loss:

$$\begin{aligned} &H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) \\ &= \max(0, 1 - y_i\{\mathsf{tr}(\mathbf{U}_+\mathbf{U}_+^T\mathbf{X}_i\mathbf{X}_i^T) - \mathsf{tr}(\mathbf{U}_-\mathbf{U}_-^T\mathbf{X}_i\mathbf{X}_i^T)\} + b) \\ &= \max(0, 1 - y_i\{\|\mathbf{U}_+^T\mathbf{X}_i\|_F^2 - \|\mathbf{U}_-^T\mathbf{X}_i\|_F^2\} + b) \end{aligned} \tag{8}$$

This expression highlights a key advantage of the bilinear classifier, namely that we never need to explicitly compute the pooled bilinear feature $\mathbf{X}_i\mathbf{X}_i^T$!

Regularization: In the hinge-loss, the parameters $\mathbf{U}_+$ and $\mathbf{U}_-$ are independent of each other. However, as noted previously, there exists a decomposition of the optimal full rank SVM in which the positive and negative subspaces are orthogonal. We thus modify the standard ℓ_2 regularization to include a positive cross-term $\|\mathbf{U}_+^T\mathbf{U}_-\|_F^2$ that favors an orthogonal decomposition [4]. This yields the final objective:

$$\begin{aligned} &\min_{\substack{\mathbf{U}_+, b \\ \mathbf{U}_-}} \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) \\ &\quad + \frac{\lambda}{2}(\|\mathbf{U}_+\mathbf{U}_+^T\|_F^2 + \|\mathbf{U}_-\mathbf{U}_-^T\|_F^2 + \|\mathbf{U}_+^T\mathbf{U}_-\|_F^2) \end{aligned} \tag{9}$$

2.2. Optimization by Gradient Descent

We call our approach the maximum Frobenius norm SVM. It is closely related to the bilinear SVM of Wolf *et al.* [33], which uses a bilinear decomposition $\mathbf{W} \approx \mathbf{U}\mathbf{V}^T$. Such non-convex bilinear models with hard rank constraints are often optimized via alternating descent [18, 29, 33, 24] or fit using convex relaxations based on the nuclear norm [13]. However, our parameterization is actually quadratic in $\mathbf{U}_+, \mathbf{U}_-$ and hence can't exploit the alternating or cyclic descent approach.

Instead, we optimize the objective function 9 using stochastic gradient descent to allow end-to-end training of both the classifier and CNN feature extractor via standard backpropagation. As discussed in the literature, model performance does not appear to suffer from non-convexity during training and we have no problems finding local minima with good test accuracy [7, 3]. The partial derivatives of our model are straightforward to compute efficiently

$$\begin{aligned} \nabla_{\mathbf{U}_+} &= 2\lambda(\mathbf{U}_+\mathbf{U}_+^T\mathbf{U}_+ + \mathbf{U}_-\mathbf{U}_-^T\mathbf{U}_+) \\ &\quad + \begin{cases} 0, & \text{if } H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) \le 0 \\ -y_i\mathbf{X}_i\mathbf{X}_i^T\mathbf{U}_+, & \text{if } H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) > 0 \end{cases} \\ \nabla_{\mathbf{U}_-} &= 2\lambda(\mathbf{U}_-\mathbf{U}_-^T\mathbf{U}_- + \mathbf{U}_+\mathbf{U}_+^T\mathbf{U}_-) \\ &\quad + \begin{cases} 0, & \text{if } H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) \le 0 \\ y_i\mathbf{X}_i\mathbf{X}_i^T\mathbf{U}_-, & \text{if } H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) > 0 \end{cases} \\ \nabla_b &= \begin{cases} 0, & \text{if } H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) \le 0 \\ -y_i, & \text{if } H(\mathbf{X}_i, \mathbf{U}_+, \mathbf{U}_-, b) > 0 \end{cases} \end{aligned} \tag{10}$$

3. Classifier Co-Decomposition for Model Compression

In many applications such as fine-grained classification, we are interested in training a large collection of classifiers and performing k-way classification. It is reasonable to expect that these classifiers should share some common structure (e.g., some feature map channels may be more or less informative for a given k-way classification task). We thus propose to further reduce the number of model parameters by performing a co-decomposition over the set of classifiers in order to isolate shared structure, similar to multi-task learning frameworks (e.g., [1]).

Suppose we have trained K Frobenius norm SVM classifiers for each of K classes. Denoting the k^{th} classifier

[3]Instead of a hard rank constraint, one could utilize the nuclear norm as a convex regularizer on $\tilde{\mathbf{W}}$. However, this wouldn't yield the computational benefits during training that we highlight here.

[4]The original ℓ_2 regularization is given by $\|\mathbf{W}\|_F^2 = \|\mathbf{U}_+\mathbf{U}_+^T - \mathbf{U}_-\mathbf{U}_-^T\|_F^2 = \|\mathbf{U}_+\mathbf{U}_+^T\|_F^2 + \|\mathbf{U}_-\mathbf{U}_-^T\|_F^2 - 2\|\mathbf{U}_+^T\mathbf{U}_-\|_F^2$ where the cross-term actually discourages orthogonality.

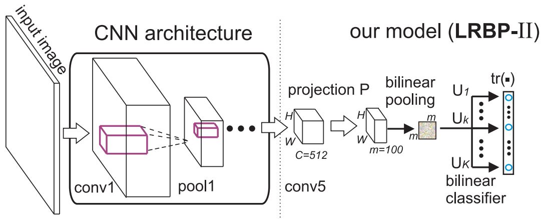

Figure 4: Another configuration of our proposed architecture that explicitly computes the bilinear pooling over co-decomposed features of lower dimension.

parameters as $\mathbf{U}_k = [\mathbf{U}_{+k}, \mathbf{U}_{-k}] \in \mathbb{R}^{c \times r}$, we consider the following co-decomposition:

$$\min_{\mathbf{V}_k, \mathbf{P}} \sum_{k=1}^{K} \|\mathbf{U}_k - \mathbf{P}\mathbf{V}_k\|_F^2, \tag{11}$$

where $\mathbf{P} \in \mathbb{R}^{c \times m}$ is a projection matrix that reduces the feature dimensionality from c to $m < c$, and $\mathbf{V}_k \in \mathbb{R}^{m \times r}$ is the new lower-dimensional classifier for the k^{th} class.

Although there is no unique solution to problem Equation 11, we can make the following statement

Proposition 2 *The optimal solution of* $\mathbf{P}$ *to Equation 11 spans the subspace of the singular vectors corresponding of the largest* m *singular values of* $[\mathbf{U}_1, \ldots, \mathbf{U}_K]$.

Therefore, without loss of generality, we can add a constraint that $\mathbf{P}$ is a orthogonal matrix without changing the value of the minimum and use SVD on the full parameters of the K classifiers to obtain $\mathbf{P}$ and $\mathbf{V}_k$'s.

In practice, we would like to avoid first learning full classifiers $\mathbf{U}_k$ and then solving for $\mathbf{P}$ and $\{\mathbf{V}_k\}$. Instead, we implement $\mathbf{P} \in \mathbb{R}^{c \times m}$ in our architecture by adding a $1 \times 1 \times c \times m$ convolution layer, followed by the new bilinear classifier layer parameterized by $\mathbf{V}_k$'s. In order to provide a good initialization for $\mathbf{P}$, we can run the CNN base architecture on training images and perform PCA on the resulting feature map activations in order to estimate a good subspace for $\mathbf{P}$. We find this simple initialization of $\mathbf{P}$ with randomly initialized $\mathbf{V}_k$'s followed by fine-tuning the whole model achieves state-of-the-art performance.

4. Analysis of Computational Efficiency

In this section, we study the computational complexity and model size in detail, and compare our model to several closely related bilinear methods, including the full bilinear model [19] and two compact bilinear models [8] by Random Maclaurin and Tensor Sketch.

We consider two variants of our proposed *low-rank bilinear pooling* (**LRBP**) architecture. In the first, dubbed *LRBP-I* and depicted in Figure 1 (d), we use the Frobenius norm to compute the classification score (see Equation 8). This approach is preferred when $hw < m$. In the second, dubbed *LRBP-II* and depicted in Figure 4, we apply the feature dimensionality reduction using $\mathbf{P}$ and then compute the pooled bilinear feature explicitly and compute the classification score according to second line of Equation 8. This has a computational advantage when $hw > m$.

Table 1 provides a detailed comparison in terms of feature dimension, the memory needed to store projection and classifier parameters, and computational complexity of producing features and classifier scores. In particular, we consider this comparison for the CUB200-2011 bird dataset [31] which has $K = 200$ classes. A conventional setup for achieving good performance of the compact bilinear model is that $d = 8,192$ as reported in [8]. Our model achieves similar or better performance using a projection $\mathbf{P} \in \mathbb{R}^{512 \times 100}$, so that $m = 100$, and using rank $r = 8$ for all the classifiers.

From Table 1, we can see that Tensor Sketch and our model are most appealing in terms of model size and computational complexity. It is worth noting that the size of our model is a hundred times smaller than the full bilinear model, and ten times smaller than Tensor Sketch. In practice, the complexity of computing features in our model $O(hwmc + hwm^2)$ is not much worse than Tensor Sketch $O(hw(c + d\log(d))$, as $m^2 \approx d$, $mc < d\log(d)$ and $m \ll c$. Perhaps the only trade-off is computation in the classification step, which is a bit higher than the compact models.

5. Experiment Evaluation

In this section, we provide details of our model implementation along with a description of baseline. We then investigate design-choices of our model, *i.e.*, the classifier rank and low-dimensional subspace determined by projection $\mathbf{P}$. Finally we report the results on four commonly used fine-grained benchmark datasets and describe several methods for generating qualitative visualizations that provide understanding of the image features driving model performance.

5.1. Implementation Details

We implemented our classifier layers within matconvnet toolbox [30] and train using SGD on a single Titan X GPU. We use the VGG16 model [28] which is pretrained on ImageNet, removing the fully connected layers, and inserting a co-decomposition layer, normalization layer and our bilinear classifiers. We use PCA to initialize $\mathbf{P}$ as described in Section 3, and randomly initialize the classifiers. We initially train only the classifiers, and then fine-tune the whole network using a batch size of 12 and a small learning rate of 10^{-3}, periodically annealed by 0.25, weight decay of 5×10^{-4} and momentum 0.9. The code and trained model

Table 1: A comparison of different compact bilinear models in terms of dimension, memory, and computational complexity. The bilinear pooled features are computed over feature maps of dimension $h \times w \times c$ for a K-way classification problem. For the VGG16 model on an input image of size 448×448 we have $h = w = 28$ and $c = 512$. The Random Maclaurin and Tensor Sketch models, which are proposed in [8] based on polynomial kernel approximation, compute a feature of dimension d. It is shown that these methods can achieve near-maximum performance with $d = 8,192$. For our model, we set $m = 100$ and $r = 8$, corresponding to the reduced feature dimension and the rank of our low-rank classifier, respectively. Numbers in brackets indicate typical values when bilinear pooling is applied after the last convolutional layer of VGG16 model over the CUB200-2011 bird dataset [31] where $K = 200$. Model size only counts the parameters above the last convolutional layer.

	Full Bilinear	Random Maclaurin	Tensor Sketch	LRBP-I	LRBP-II
Feature Dim	c^2 [262K]	d [10K]	d [10K]	mhw [78K]	m^2 [10K]
Feature computation	$O(hwc^2)$	$O(hwcd)$	$O(hw(c + d \log d))$	$O(hwmc)$	$O(hwmc + hwm^2)$
Classification comp.	$O(Kc^2)$	$O(Kd)$	$O(Kd)$	$O(Krmhw)$	$O(Krm^2)$
Feature Param	0	$2cd$ [40MB]	$2c$ [4KB]	cm [200KB]	cm [200KB]
Classifier Param	Kc^2 [KMB]	Kd [K·32KB]	Kd [K·32KB]	Krm [K·3KB]	Krm [K·3KB]
Total ($K = 200$)	Kc^2 [200MB]	$2cd + Kd$ [48MB]	$2c + Kd$ [8MB]	$cm + Krm$ [0.8MB]	$cm + Krm$ [0.8MB]

have been released to the public[5].

We find that proper feature normalization provides a non-trivial improvement in performance. Our observation is consistent with the literature on applying normalization to deal with visual burstiness [11, 19]. The full bilinear CNN and compact bilinear CNN consistently apply sign square root and ℓ_2 normalization on the bilinear features. We can apply these normalization methods for our second configuration (described in Section 4). For our first configuration, we don't explicitly compute the bilinear feature maps. Instead we find that sign square root normalization on feature maps at $conv5\_3$ layer results in performance on par with other bilinear pooling methods while additional ℓ_2 normalization harms the performance.

5.2. Configuration of Hyperparameters

Two hyperparameters are involved in specifying our architecture, the dimension m in the subspace determined by $\mathbf{P} \in \mathbb{R}^{c \times m}$ and the rank r of the classifiers $\mathbf{V}_k \in \mathbb{R}^{m \times r}$ for $k = 1, \ldots, K$. To investigate these two parameters in our model, we conduct an experiment on CUB-200-2011 bird dataset [31], which contains 11,788 images of 200 bird species, with a standard training and testing set split. We do not use any part annotation or masks provided in the dataset.

We first train a full-rank model on the bilinear pooled features and then decompose each classifier using eigenvalue decomposition and keep the largest magnitude eigenvalues and the corresponding vectors to produce a rank-r classifier. After obtaining low-rank classifiers, we apply co-decomposition as described in Section 3 to obtain projector $\mathbf{P}$ and compact classifiers $\mathbf{V}_k$'s. We did not perform fine-tuning of these models but this quick experiment provides a good proxy for final model performance over a range of architectures.

We plot the classification accuracy vs. rank r and reduced dimension m (rDim) in Figure 5, the average reconstruction fidelity measured by peak signal-to-noise ratio to the original classifier parameters $\mathbf{U}_k$ versus rank r and dimension m in Figure 6, and model size versus rank r and dimension m in Figure 7.

As can be seen, the reconstruction fidelity (measured in the peak signal-to-noise ratio) is a good guide to model performance prior to fine tuning. Perhaps surprisingly, even with $r = 8$ and $m = 100$, our model achieves near-maximum classification accuracy on this dataset (Figure 5) with model parameters compressed by a factor of 100 over the full model (Figure 7). Based on this analysis, we set $r = 8$ and $m = 100$ for our quantitative benchmark experiments.

5.3. Baseline Methods

We use VGG16 [28] as the base model in all comparison to be consistent with previous work [19, 8].

Fully Connected layers (FC-VGG16): We replace the last fully connected layer of VGG16 base model with a randomly initialized K-way classification layer and fine-tune. We refer this as "FC-VGG16" which is commonly a strong baseline for a variety of computer vision tasks. As VGG16 only takes input image of size 224×224, we resize all inputs for this method.

Improved Fisher Encoding (Fisher): Fisher encoding [22] has recently been used as an encoding and pooling alternative to the fully connected layers [5]. Consistent with [8, 19], we use the activations at layer $conv5\_3$ (prior to ReLU) as local features and set the encoding to use 64 GMM components for the Fisher vector representation.

Full Bilinear Pooling (Full Bilinear): We use full bilinear pooling over the $conv5\_3$ feature maps (termed "symmetric structure" in [19]) and apply element-wise sign

[5]`https://github.com/aimerykong/Low-Rank-Bilinear-Pooling`

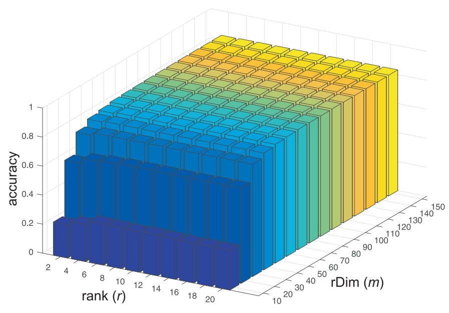

Figure 5: Classification accuracy on CUB-200 dataset [31] vs. reduced dimension (m) and rank (r).

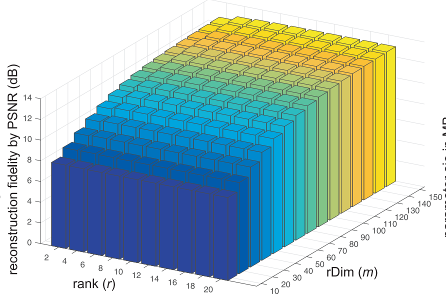

Figure 6: Reconstruction fidelity of classifier parameters measured by peak signal-to-noise ratio.

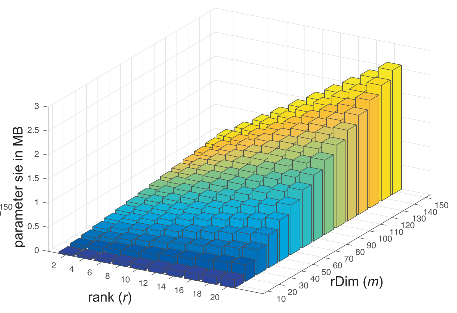

Figure 7: The learned parameter size versus reduced dimension (m) and rank (r).

square root normalization and ℓ_2 normalization prior to classification.

Compact Bilinear Pooling: We report two methods proposed in [8] using Random Maclaurin and Tensor Sketch. Like Full Bilinear model, element-wise sign square root normalization and ℓ_2 normalization are used. We set the projection dimension $d = 8,192$, which is shown to be sufficient for reaching close-to maximum accuracy [8]. For some datasets, we use the code released by the authors to train the model; otherwise we display the performance reported in [8].

5.4. Quantitative Benchmarking Experiment

We compare state-the-art methods on four widely used fine-grained classification benchmark datasets, CUB-200-2011 Bird dataset [31], Aircrafts [21], Cars [17], and describing texture dataset (DTD) [4]. All these datasets provide fixed train and test split. We summarize the statistics of datasets in Table 3. In training all models, we only use the category label without any part or bounding box annotation provided by the datasets. We list the performance of these methods in Table 2 and highlight the parameter size of the models trained on CUB-200 dataset in the last row.

From the comparison, we can clearly see that Fisher vector pooling not only provides a smaller model than FC-VGG16, but also consistently outperforms it by a notable margin. All the bilinear pooling methods, including ours, achieve similar classification accuracy, outperforming Fisher vector pooling by a significant margin on these datasets except DTD. However, our model is substantially more compact than the other methods based on bilinear features. To the best of our knowledge, our model achieves the state-of-the-art performance on these datasets without part annotation [10, 15], and even outperforms several recently proposed methods trained that use supervised part annotation [37]. Although there are more sophisticated methods in literature using detailed annotations such as parts or bounding box [36, 35], our model relies only on the category label. These advantages make our model appealing not only for memory-constrained devices, but also in weakly supervised fine-grained classification in which detailed part annotations are costly to obtain while images with category label are nearly free and computation during model training becomes the limiting resource.

Note that the simple PCA reduced feature with full bilinear pooling gives a large model reduction without noticeable accuracy loss [19]. We provide a comparison on CUB-2011-200 dataset in Table 4 *w.r.t* the feature size (Fea. Dim.), projection matrix (Feat. Param.), classifier size (Cls. Param.) and accuracy. Feature computation and classification for the listed methods are linear in feature size, so our method and PCA with 16$\times$ reduction require similar computation. However, our method enjoys a further ten times reduction in model size and performs better than straight PCA on the feature map. Moreover, our co-decomposition addresses scaling to large numbers of categories.

5.5. Qualitative Visualization

To better understand our model, we adopt three different approaches to visualizing the model response for specific input images. In the first method, we feed an input image to the trained model, and compute responses $\mathbf{Y} = [\mathbf{U}_{+1}, \mathbf{U}_{-1}, \dots, \mathbf{U}_{+k}, \mathbf{U}_{-k}, \dots, \mathbf{U}_{+K}, \mathbf{U}_{-K}]^T\mathbf{X}$ from the bilinear classifier layer. Based on the ground-truth class label, we create a modified response $\bar{\mathbf{Y}}$ by zeroing out the part corresponding to negative Frobenius score $(-\|\mathbf{U}_-^T\mathbf{X}\|_F^2)$ for the ground-truth class, and the part to the positive Frobenius scores $(\|\mathbf{U}_+^T\mathbf{X}\|_F^2)$ in the remaining classifiers, respectively. This is similar to approaches used for visualizing HOG templates by separating the positive and negative components of the weight vector. To visualize the result, we treat $\bar{\mathbf{Y}}$ as the target and backpropagate the difference to the input image space, similar to [27]. For the second visualization, we compute the magnitude of feature activations averaged across feature channels used by the bilinear classifier. Finally, we produce a third visualization by repeatedly remove superpixels from the input image, se-

Table 2: Classification accuracy and parameter size of: a fully connected network over VGG16 [28], Fisher vector [5], Full bilinear CNN [19], Random Maclaurin [8], Tensor Sketch [8], and our method. We run Random Maclaurin and Tensor Sketch with the code provided in [8] with their conventional configuration (*e.g.* projection dimension $d = 8192$).

	FC-VGG16	Fisher	Full Bilinear	Random Maclaurin	Tensor Sketch	LRBP (Ours)
CUB [31]	70.40	74.7	84.01	83.86	84.00	**84.21**
DTD [4]	59.89	65.53	64.96	65.57	64.51	**65.80**
Car [17]	76.80	85.70	91.18	89.54	90.19	**90.92**
Airplane [21]	74.10	77.60	87.09	87.10	87.18	**87.31**
param. size (CUB)	67MB	50MB	200MB	48MB	8MB	0.8MB

Table 3: Summary statistics of datasets.

	# train img.	# test img.	# class
CUB [31]	5994	5794	200
DTD [4]	1880	3760	47
Car [17]	8144	8041	196
Airplane [21]	6667	3333	100

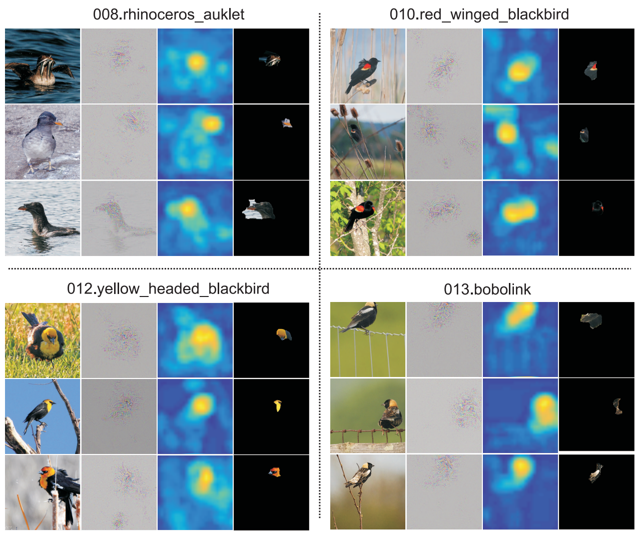

Figure 8: (Best seen in color.) Each of the four panels show different bird species; the four columns display the input images and the visualization maps using three different methods as described in Section 5.5. We can see our model tends to ignore features in the cluttered background and focus on the most distinct parts of the birds.

lecting the one that introduces minimum drop in classification score This is similar to [25, 38]. In Figure 8, we show some randomly selected images from four different classes in CUB-200-2011 dataset and their corresponding visualizations.

The visualizations all suggest that the model is capable of ignoring cluttered backgrounds and focuses primarily on the bird and even on specific discriminative parts of each bird. Moreover, the highlighted activation region changes w.r.t the bird size and context, as shown in the first panel of Figure 8. For the species "010.red_winged_blackbird", "012.yellow_headed_blackbird" and "013.bobolink", the most distinctive parts, intuitively, are the red wings, yellow head and neck, and yellow nape, respectively. Our model naturally appears to respond to and localize these parts. This provides a partial explanation as to why simple global pooling achieves such good results without an explicit spatial transformer or cross-channel pooling architecture (e.g. [20])

Table 4: Comparison to PCA reduced versions of full bilinear pooling on CUB-2011-200 dataset.

	64×	32×	16×	ours
Feat. Dim. (K)	32	16	8	10
Feat. Param. (MB)	0.13	0.06	0.03	0.20
Cls. Param. (MB)	25.00	12.50	6.25	0.61
Accuracy (%)	84.18	83.85	83.81	84.21

6. Conclusion

We have presented an approach for training a very compact low-rank classification model that is able to leverage bilinear feature pooling for fine-grained classification while avoiding the explicit computation of high-dimensional bilinear pooled features. Our Frobenius norm based classifier allows for fast evaluation at test time and makes it easy to impose hard, low-rank constraints during training, reducing the degrees of freedom in the parameters to be learned and yielding an extremely compact feature set. The addition of a co-decomposition step projects features into a shared subspace and yields a further reduction in computation and parameter storage. Our final model can be initialized with a simple PCA step followed by end-to-end fine tuning.

Our final classifier model is one to two orders of magnitude smaller than existing approaches and achieves state-of-the-art performance on several public datasets for fine-grained classification by using only the category label (without any keypoint or part annotations). We expect these results will form a basis for future experiments such as training on weakly supervised web-scale datasets [16], pooling multiple feature modalities and further compression of models for use on mobile devices.

Acknowledgement

This project is supported by NSF grants IIS-1618806, IIS-1253538, DBI-1262547 and a hardware donation from NVIDIA.

References

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[2] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *Computer Vision–ECCV 2012*, pages 430–443. Springer, 2012.

[3] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

[4] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[5] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3828–3836, 2015.

[6] Y. Cui, F. Zhou, Y. Lin, and S. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. *arXiv preprint arXiv:1512.05227*, 2015.

[7] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[8] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. *CVPR*, 2016.

[9] S. Huang, Z. Xu, D. Tao, and Y. Zhang. Part-stacked cnn for fine-grained visual categorization. *arXiv preprint arXiv:1512.08086*, 2015.

[10] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2008–2016, 2015.

[11] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1169–1176. IEEE, 2009.

[12] P. Kar and H. Karnick. Random feature maps for dot product kernels. In *AISTATS*, volume 22, pages 583–591, 2012.

[13] T. Kobayashi. Low-rank bilinear classification: Efficient convex optimization and extensions. *International Journal of Computer Vision*, 110(3):308–327, 2014.

[14] P. Koniusz and A. Cherian. Sparse coding for third-order super-symmetric tensor descriptors with application to texture recognition. *CVPR*, 2016.

[15] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015.

[16] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. *arXiv preprint arXiv:1511.06789*, 2015.

[17] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[18] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[19] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[20] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4749–4757, 2015.

[21] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[22] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision*, pages 143–156. Springer, 2010.

[23] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247. ACM, 2013.

[24] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In *Advances in neural information processing systems*, pages 1482–1490, 2009.

[25] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *KDD*, 2016.

[26] M. Simon, E. Rodner, and J. Denzler. Part detector discovery in deep convolutional neural networks. In *Computer Vision–ACCV 2014*, pages 162–177. Springer, 2014.

[27] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.

[30] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

[31] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[32] Y. Wang, J. Choi, V. I. Morariu, and L. S. Davis. Mining discriminative triplets of patches for fine-grained classification. *CVPR*, 2016.

[33] L. Wolf, H. Jhuang, and T. Hazan. Modeling appearances with low-rank svm. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE, 2007.

[34] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

[35] H. Zhang, T. Xu, M. Elhoseiny, X. Huang, S. Zhang, A. Elgammal, and D. Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *CVPR*, 2016.

[36] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[37] N. Zhang, E. Shelhamer, Y. Gao, and T. Darrell. Fine-grained pose prediction, normalization, and recognition. In *ICLR workshop*, 2016.

[38] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.