Bilinear CNN Models for Fine-grained Visual Recognition

Tsung-Yu Lin Aruni RoyChowdhury Subhransu Maji

Abstract—We present bilinear CNNs, an architecture that efficiently represents an image as a pooled outer product of two CNN features, that is effective at fine-grained recognition tasks. These models capture localized part-feature interactions similar to those in part-based models, but can also be seen as an orderless texture representation. Based on this observation we derive a family of end-to-end trainable bilinear models that generalize classical image representations, such as the second-order pooling, Fisher-vectors, vector-of-locally-aggregated descriptors, and bag-of-visual-words. This allows domain-specific fine-tuning and visualization of the learned models by approximate inversion. Through a number of experiments we show that these models offer better accuracy, speed, and memory trade-offs compared to prior work on various fine-grained, texture, and scene recognition datasets. The source code for the complete system is available at http://vis-www.cs.umass.edu/bcnn.

Index Terms—Fine-grained recognition, Texture recognition, Bilinear models, Convolutional Neural Networks, Visualization

1 INTRODUCTION

Fine-grained recognition tasks typically involve discrimination among categories that have a shared structure but differ in subtle ways, e.g. distinguishing between a “California gull” and a “Ringed-bill Gull”. This requires recognition of highly localized attributes under changes in pose, viewpoint, lighting, and other factors. While most humans can recognize these attributes, they often require a field-guide to identify which ones are relevant for a given category. Thus, accurate computer vision systems can significantly reduce the manual effort required for these tasks.

Currently, there are two broad categories of computer vision techniques that are effective for these tasks. Explicit part-based models construct representations by localizing parts of an object and extracting features conditioned on their detected locations. This makes subsequent reasoning about categories easier since the appearance variations due to location, pose, and viewpoint changes are factored out. Holistic models on the other hand construct a representation of the entire image directly. These include bag-of-visual-word (BoVW) representations, and their variants such as Fisher-vector representation [41] and vector-of-locally-aggregated descriptors (VLAD) [27], popularized for texture analysis. Other examples are histogram-of-oriented-gradients representations [11] and those based on multi-layer convolutional neural networks (CNNs) [31], [32]. While holistic representations do not have an explicit part-detection step, they may have an implicit representation of parts. For example, visual words in texture representations, or features that emerge in higher layers of a CNN hierarchy have been shown to correlate with semantic parts [58]. However, the semantic alignment of these parts is not as clearly defined as in explicit part-based models.

While recent results have shown the benefits of both explicit part-based methods and holistic representations, neither offers an complete solution. Explicit part-based models are more accurate and tend to generalize better when training data is limited, but require part annotations. This makes them less scalable to domains where such annotations may be difficult to obtain, such as categories without a clearly defined set of parts such as buildings, trees, or images in biomedical domains.

In this paper we argue that the main benefit of part-based models is that the resulting representation is translation and pose invariant. Texture representations are translationally invariant by design since they are based on orderless aggregation of features in an image. Recently, texture representations constructed from features corresponding to layers of a CNN have been shown to be very effective at fine-grained and texture recognition tasks [9], [22]. In such representations, local features are often encoded to a high-dimensional space before aggregation. The encoding step is typically not differentiable, hence most texture representations, and their recent deep extensions, fix the local representation and only learn the parameters of the encoder for each dataset. Thus the full potential of texture representations has not been reached due to the difficulty of combining texture representations with existing deep learning pipelines.

We present bilinear CNNs that address several drawbacks of existing part-based and texture representations. Our key insight is that several widely-used texture encoders can be written as an outer product of two suitably designed features. This results in a representation that is linear in the two features, hence the name bilinear models. The simplest bilinear model is one which consists of two independent features combined with an outer product. A variant is one where the two features are identical and is related to the second-order pooling (O2P) representation [5]. We also show that BoVW, Fisher vector and VLAD representations can be written as bilinear models. Furthermore, we analyze simplified variants of these models for which gradients have a simpler form resulting in a family of differentiable

- T.-Y. Lin, A. RoyChowdhury, and S. Maji are with the College of Information and Computer Sciences, University of Massachusetts Amherst, USA. E-mails: {tsungyulin, arunirc, smaji}@cs.umass.edu
encoding layers which are general-purpose, and can be plugged into existing CNN architectures for end-to-end training. Remarkably, this has significant benefits and allows bilinear CNN models to even outperform part-based CNN models on a variety of fine-grained recognition datasets, even though they are trained with image labels only.

The manuscript combines the analysis of our earlier works [34], [35] and extends them in a number of ways. We compare our results with deep variants of Fisher vector, VLAD [1], and BoVW representations, that are end-to-end trainable, on a variety of datasets and study the effect of domain-specific fine-tuning. We extend the visualization technique in [38] to include results using various texture representations. We extend the analysis of dimensional reductions techniques and provide trade-off curves between accuracy and the dimensionality for different models, including a direct comparison with the recently proposed “compact bilinear pooling” technique [17].

2 BILINEAR MODELS FOR IMAGE CLASSIFICATION

A bilinear model $B$ for image classification consists of a quadruple $B = \{f_A, f_B, \mathcal{P}, \mathcal{C}\}$. Here $f_A$ and $f_B$ are feature functions, $\mathcal{P}$ is a pooling function and $\mathcal{C}$ is a classification function. A feature function is a mapping $f : \mathcal{L} \times \mathcal{I} \rightarrow \mathbb{R}^{c \times D}$ that takes an image $\mathcal{I}$ and a location $\mathcal{L}$ and outputs a feature of size $c \times D$. We refer to locations generally which can include position and scale. The feature outputs are combined at each location using the matrix outer product, i.e., the bilinear feature combination of $f_A$ and $f_B$ at a location $l$ is given by

$$\text{bilinear}(l, \mathcal{I}, f_A, f_B) = f_A(l, \mathcal{I})^T f_B(l, \mathcal{I}).$$

Both $f_A$ and $f_B$ must have the same feature dimension $c$ to be compatible. To obtain an image descriptor, the pooling function $\mathcal{P}$ aggregates the bilinear feature across all locations in the image. One choice of pooling is to simply sum all the bilinear features, i.e., $\phi(\mathcal{I}) = \sum_{l \in \mathcal{L}} \text{bilinear}(l, \mathcal{I}, f_A, f_B)$. An alternative is max-pooling. Both of these ignore the location of the features and are hence orderless. If $f_A$ and $f_B$ extract features of size $C \times M$ and $C \times N$ respectively, then $\phi(\mathcal{I})$ is of size $M \times N$. The bilinear vector, obtained by reshaping $\phi(\mathcal{I})$ to size $MN \times 1$, is a general purpose image descriptor that can be used with a classification function $\mathcal{C}$ (Fig. 1). Intuitively, the bilinear form allows the outputs of the feature extractors $f_A$ and $f_B$ to be conditioned on each other by considering their pairwise interactions similar to a quadratic kernel expansion.

### 2.1 Feature functions

A natural candidate for the feature function $f$ is a CNN consisting of a hierarchy of convolutional and pooling layers. In our experiments we use CNNs pre-trained on the ImageNet dataset [47] truncated at an intermediate layer as feature functions. By pre-training we benefit from additional training data when domain specific data is scarce. This has been shown to be beneficial for a number of tasks ranging from object detection, texture recognition, to fine-grained classification [8], [12], [21], [45]. Another advantage of using CNNs, is the resulting network can process images of an arbitrary size and produce outputs indexed by image location and feature channel.

Our earlier work [35] experimented with models where the feature functions $f_A$ and $f_B$ were either independent or fully shared shared. Here, we also experiment with feature functions that share a part of the feed-forward computation as seen in Fig. 3. The low-dimensional approximations of bilinear CNN representations we present in Sec. 6.1, as well as the feature functions used to approximate classical texture representations we present in Sec. 3 have this form.

### 2.2 Pooling, normalization, and classification

In our experiments we use sum pooling to aggregate the bilinear features across the locations in an image. In addition we perform a normalization step where the bilinear vector $x = \phi(\mathcal{I})$ is passed through a signed square-root ($y \leftarrow \text{sign}(x) \sqrt{|x|}$), followed by $\ell_2$ normalization ($z \leftarrow y / ||y||_2$) inspired by [42]. This improves performance in practice (see the appendix of our earlier work [35] for experiments evaluating the effect of these steps). For the classification function $\mathcal{C}$ we use logistic regression or linear SVM. This can be replaced with a multi-layer neural network if non-linearity is desirable.

### 2.3 End-to-end training

Since the overall architecture is a directed acyclic graph, the parameters can be trained by back-propagating the gradients of the classification loss (e.g., conditional log-likelihood). The bilinear form simplifies the gradients at the pooling layer. If the outputs of the two networks are matrices $A$ and $B$ of size $L \times M$ and $L \times N$ respectively, then the pooled bilinear feature is $x = AB^T$ of size $M \times N$. Let $\frac{d\ell}{dx}$ be the gradient of the loss function $\ell$ wrto. $x$, then by chain rule of gradients we have:

$$\frac{d\ell}{dA} = B \left( \frac{d\ell}{dx} \right)^T, \quad \frac{d\ell}{dB} = A \left( \frac{d\ell}{dx} \right).$$

Thus, as long as the features $A$ and $B$ are differentiable wrto. the model parameters, the entire model can be trained end-to-end. The scheme is illustrated in Fig 2.
variants that are end-to-end trainable. To simplify further

Fig. 2. Computing gradients in the bilinear CNN model.

3 TEXTURE DESCRIPTORS AS BILINEAR MODELS

In this section we show that various orderless texture de-
scriptors can be written in the bilinear form and derive variants that are end-to-end trainable. To simplify further analysis, we will decompose the feature function $f$ as $f(l, I) = g(h(l, I)) = g(x)$ to denote the explicit dependency on the image and the location of $h$ and additional non-linearities $g$.

Since the properties of texture are usually translationally invariant, most texture representations are based on orderless aggregation of local image features, e.g. global averaging. Some non-linear encoding is applied before aggregation of local features to improve robustness and discriminative-ness of the representation. Thus texture representations can be defined by the choice of the local features, the encoding function, and the pooling function.

One of the earliest texture representation was the bag-of-visual-words (BoVW) [10] representation. It was shown to be effective at several recognition tasks beyond texture. While variants differ on how the visual words are learned and represented, a popular approach is to model the distribution of features using a Gaussian mixture model (GMM) with $k$ components. In this case the encoder assigns a feature $x$ to the $k$ components based on its GMM posteriors. These encoded descriptors are averaged across the image to obtain the BoVW representation. Using our notation, if we set $g_A(x) = 1, \forall x$, and $\eta(x)$ to be $g_B(x)$, the GMM posterior of the feature $x$, the BoVW model can be written as a bilinear model $(1, \eta(x), P, C)$.

The vector-of-locally-aggregated descriptor (VLAD) [27] encodes a descriptor $x$ as $(x - \mu_k) \odot \eta(x)$, where $\odot$ is the kronecker product and $\mu_k$ is the closest center to $x$. In the VLAD model, $\eta(x)$ is set to one for the closest center and zero elsewhere, also referred to as “hard assignment.” These encodings are aggregated across the image by sum pooling. Thus VLAD can be written as a bilinear model with $g_A(x) = [x - \mu_1; x - \mu_2; \ldots; x - \mu_k]$. Here, $g_A$ has $k$ rows each corresponding to a center. And $g_B(x) = \text{diag}(\eta(x))$, a matrix with $\eta(x)$ in the diagonal and zero elsewhere. Notice that the feature functions for VLAD output a matrix with $k > 1$ rows at each location.

The Fisher-vector (FV) [42] representation computes both the first order $\alpha_i = \Sigma_i^{-2}(x - \mu_i)$ and second order $\beta_i = \Sigma_i^{-1}(x - \mu_i) \odot (x - \mu_i) - 1$ statistics, which are aggregated weighted by the GMM posteriors $\eta(x)$. Here $\mu_i$ and $\Sigma_i$ are the mean and covariance of the $i^{th}$ GMM component respectively and $\odot$ denotes element-wise multiplication. This can be written as a bilinear model with $g_A = [\alpha_1 \beta_1; \alpha_2 \beta_2; \ldots; \alpha_k \beta_k]$ and $g_B = \text{diag}(\eta(x))$.

The second-order pooling (O2P) technique [5] was shown to be effective for semantic segmentation. The method constructs representation by computing the covariance statistics of SIFT features within a region, followed by non-linearities. Thus O2P can be written as a bilinear model with $f_A = f_B = f_{\text{soft}}$.

The appearance-based cluster centers learned by the encoder $\eta(x)$ in the BoVM, VLAD and FV representations can be thought of as part detectors. Indeed it has been observed that the GMM centers tend to localize facial landmarks when trained on faces [40]. Thus, by encoding the joint statistics of the encoding $\eta(x)$ and the appearance $x$, the models can effectively describe appearance of parts regardless of where they appear in the image. This is particularly useful for fine-grained recognition where objects are not localized in the image.

3.1 End-to-end trainable formulations

Prior work on combining CNN-based representations with texture encoders have not been trained end-to-end since the encoding layer is not easily differentiable. In this section we derive variants, and in some cases simplifying them, for which gradients have a simple form. These differentiable encoding layers can be plugged into existing deep architectures yielding a family of models that enjoy the benefits of features extracted from pre-trained models, the benefits offered by texture representations, as well as the ability to perform domain-specific fine-tuning. In our experiments we find that the ability to directly fine-tune these models leads to significant improvements in accuracy across a variety of fine-grained datasets.

Recently, an end-to-end trainable formulation of VLAD as proposed [1] by replacing the “hard assignment” $\eta(x)$ in $g_B$ by a differentiable “soft assignment” $\tilde{\eta}(x)$. Given the $k$-th cluster center $\mu_k$, the $k$-th component of the soft assignment vector for an input $x$ is given by,

$$\tilde{\eta}_k(x) = \frac{e^{-\gamma||x - \mu_k||^2}}{\sum_{k'} e^{-\gamma||x - \mu_k||^2}} = \frac{e^{w_k^T x + b_k}}{\sum_{k'} e^{w_k^T x + b_{k'}}} \quad (3)$$

where $w_k = 2\gamma\mu_k$, $b_k = -\gamma||\mu_k||^2$ and $\gamma$ is a parameter of the model. This is simply the softmax operation applied after a convolution layer with a bias term, and can be implemented using standard CNN building blocks. The function $g_A$ remains unchanged [$x - \mu_1; x - \mu_2; \ldots; x - \mu_k$].

We extend the VLAD approximation to the FV model by appending the second-order differences to the feature $g_A$ in the VLAD formulation, i.e. $g_A = [x - \mu_1, (x - \mu_1)^2; x - \mu_2, (x - \mu_2)^2; \ldots; x - \mu_k, (x - \mu_k)^2]$. Here, the squaring operation is done in an element-wise fashion, i.e. $(x - \mu_i)^2 = (x - \mu_i) \odot (x - \mu_i)$. We also discard the covariances and prior terms present in the true GMM posterior in the FV model in this approximation and instead use the softmax approximation $g_B = \text{diag}(\tilde{\eta}(x))$, identical to the formulation for the VLAD approximation.

For the BoVW approximation, the soft assignments $\tilde{\eta}(x)$ are taken to correspond to the GMM posteriors in the original BoVW formulation. We then sum-pool over all the spatial locations by setting $g_A = \tilde{\eta}(x)$ and $g_B = 1$.

In these approximations the centers $\mu = [\mu_1, \mu_2, \ldots, \mu_k]$ can be initialized using GMMs or k-means. However, being
trainable parameters these can be directly updated along with the rest of the network during training. Prior work has either held these fixed (e.g. Fisher vector CNN [8]), or resorted to indirect training by fine-tuning the standard CNN with fully-connected layers and then constructing Fisher vector representations on fine-tuned CNN features. Our experiments show a clear advantage of direct fine-tuning (Sec. 5.2).

4 Related work

Fine-grained recognition using CNNs After AlexNet’s [31] impressive performance on ImageNet classification challenge was revealed, several authors [12], [45] demonstrated that features extracted from layers of the CNN are also effective at fine-grained recognition tasks. Building on prior work on part-based techniques [2], [13], [60], Zhang et al. [39], and Branson et al. [3] showed the benefits of combining CNN-based part detectors and CNN-based features for fine-grained tasks.

Among the non part-based techniques, texture descriptors such as Fisher vectors and VLAD have traditionally been effective for fine-grained recognition. For example, the top performing method on FGCOMP’12 challenge used SIFT-based Fisher-vector representation [23]. Their deep variants [9] have also been shown to be more effective than standard CNNs on several tasks.

Recent improvements in CNN architectures have also resulted in improvements in fine-grained recognition. This includes architectures that have increased depth such as the “very deep” architectures [51] from the Oxford’s VGG group, inception architectures [52], and “ultra deep” residual networks [24]. Other techniques have shown improvements by augmenting CNNs with “attention” mechanisms that allow focused reasoning on parts of an image. Spatial transformer network [26] extend CNNs with parameterized image transformations have been effective at fine-grained recognition tasks. Bilinear CNNs can be viewed as an implicit spatial attention model since the outer product captures multiplicative feature interactions similar to those in an attention mechanism.

Texture representations Texture representations have been widely studied for decades. Early work [33] captures the texture appearance by computing the statistics of linear filter-bank responses. Cimpoi et al. [9] replaced the linear filter-bank with nonlinear CNN filters to build Fisher vector representations and showed significant improvements over prior results on texture, material, and scene recognition datasets. Concurrent to our work, Gatys et al. [18], [19] showed that the Gram-matrix representation constructed from multiple layers of a CNN is an effective texture representation and can be used to generate new textures or stylize images. The Gram-matrix representation is the identical to the bilinear-vector representation when the features \(f_A\) and \(f_B\) are identical.

Polynomial kernels and sum-product networks An alternate strategy for combing features from two networks is to concatenate them and learn their pairwise interactions through a series of layers on top. However, doing this naively requires a large number of parameters since there are \(O(n^2)\) interactions over \(O(n)\) features requiring a layer with \(O(n^3)\) parameters. Our explicit representation using an outer product has no parameters and is similar to a quadratic kernel expansion used in kernel support vector machines [48]. However, one might be able to achieve similar approximations using alternate architectures such as sum-product networks that efficiently model multiplicative interactions [20].

Bilinear model variants Bilinear models were proposed by Tanenbaum and Freeman [53] to model two-factor variations such as “style” and “content” for images. While we also model two factor variations in location and appearance of parts, our goal is classification and not the explicit
modeling of these factors. Our work is related to bilinear classifiers [43] that express the classifier as a product of two low-rank matrices. However, in our model the features are bilinear, while the classifier itself is linear. Our models based on low dimensional representations described in Sect. 6.1 can be interpreted as bilinear classifiers.

Our model is related to “two-stream” architectures used to analyze videos where one network models the temporal aspect, while the other models the spatial aspect [15], [50]. The idea of combining two features using the outer product has also been shown to be effective for other tasks such as visual question-answering [16] where text and visual features are combined, action recognition [14] where flow and image features are combined.

Scalability and speed Bilinear CNNs compare favorably with other CNN architectures in terms of speed. Our MatConvNet-based [56] implementation runs between 30 to 100 frames per second on a NVIDIA Titan X GPU with cudnn-v5 depending on the model architecture. Even with faster object detection modules such as Faster R-CNNs [46], Single-Shot Detector (SSD) [37], part-based models for fine-grained recognition are 2-10× slower. The main advantage is that our models require image labels only and hence can be easily applied to different fine-grained datasets.

5 IMAGE CLASSIFICATION EXPERIMENTS

In this section, we present experiments on various fine-grained, texture, and indoor-scene recognition datasets. We describe the experimental setup in Sec. 5.1 and provide a detailed comparison of the accuracy and the effect of fine-tuning on various fine-grained recognition (Sec. 5.2), texture, and scene-recognition datasets (Sec. 5.3). Our experiments are conducted using NVIDIA Titan X GPU with MatConvNet [56] and VLFEAT [55] libraries.

5.1 Methods

The methods in our experiments are based on combining various features extractors with various pooling encoders, and their end-to-end trainable variants. These are described as follows:

**Fisher vector with SIFT (FV-SIFT)** We implemented a FV baseline using dense SIFT features [42] extracted using VLFEAT [55]. The input image is first resized to 448×448 before SIFT features with binsize of 8 pixels are computed densely across the image with a stride of 4 pixels. The features are PCA projected to 80 dimensions before learning a GMM with 256 components.

**CNN with fully-connected layers (FC-CNN)** This is based on the features extracted from the last fully-connected layer before the softmax layer of the CNN. The input image is resized to 224×224 (the input size of the CNN) and mean-subtracted before propagating it through the CNN. For fine-tuning we replace the 1000-way classification layer trained on ImageNet dataset with a k-way softmax layer where k is the number of classes in the fine-grained dataset. The parameters of the softmax layer are initialized randomly and we continue training the network on the dataset for several epochs at a smaller learning rate while monitoring the validation error. Once the networks are trained, the layer before the softmax layer is used to extract features.

**Bilinear CNN model (B-CNN)** This refers to the class of bilinear models proposed in our earlier work [35] that consist of a direct outer product of CNN features. When the two CNNs are identical, the model can also be thought of as an extension of O2P using deep features. We consider several bilinear CNN models – (i) initialized with two identical VGG-M networks truncated at the relu5 layer denoted by B-CNN [M,M], (ii) initialized with a VGG-D and VGG-M network truncated at the relu5 layer respectively denoted by B-CNN [D,M], and (iii) initialized with two identical VGG-D networks truncated at the relu5 layer denoted by B-CNN [D,D]. Identical to the setting in FV-CNN, the input images are first resized to 448×448 and features are extracted using the two networks before bilinear
combination, sum-pooling, and normalization. The VGG-D network produces a slightly larger output 28×28 compared to 27×27 of the VGG-M network. We simply downsample the output of the D-Net by ignoring a row and column. The pooled bilinear feature is of size 512×512, which comparable to that of FV-CNN (512×128) and FV-SIFT (80×512). For fine-tuning we add a k-way softmax layer. We adopt the two step training procedure of [3] where we first train the last layer using logistic regression, a convex optimization problem, followed by fine-tuning the entire model using back-propagation for several epochs (about 45 – 100 depending on the dataset and model) at a relatively small learning rate (η = 0.001). Across the datasets we found the hyperparameters for fine-tuning were fairly consistent.

5.1.1 SVM training and evaluation

In all our experiments, training and validation sets are combined and one-vs-all linear SVMs on the extracted features are trained by setting the learning hyperparameter $C_{\text{svm}} = 1$. Since our features are $\ell_2$ normalized, the optimal of $C_{\text{svm}}$ is likely to be independent of the dataset. The trained classifiers are calibrated by scaling the weight vector such that the median scores of positive and negative training examples are at +1 and −1 respectively. For each dataset we double the training data by flipping images and at test time we average the predictions of the image and its flipped copy and assign the class with the highest score. Directly using the softmax predictions results in a slight drop in accuracy compared to linear SVMs. Performance is measured as the percentage of correct image predictions for all datasets.

5.2 Fine-grained recognition

We evaluate methods on following fine-grained datasets and report the per-image accuracy in Table 2.  

CUB-200-2011 [57] dataset contains 11,788 images of 200 bird species which are split into roughly equal train and test sets with detail annotation of parts and bounding boxes. As birds appear in different poses and viewpoints and occupy small portion of image in cluttered background, classifying bird species is challenging. Notice that in all our experiments, we only use image labels during training without any part or bounding box annotation. In the following sections, “birds” refers to the results on this dataset. 

FGVC-aircraft dataset [39] consists of 10,000 images of 100 aircraft variants, and was introduced as a part of the FGComp 2013 challenge. The task involves discriminating variants such as the Boeing 737-300 from Boeing 737-400. The differences are subtle, e.g., one may be able to distinguish them by counting the number of windows in the model. Unlike birds, airplanes tend to occupy a significantly larger portion of the image and appear in relatively clear background. Airplanes also have a smaller representation in the ImageNet dataset compared to birds.

Cars dataset [30] contains 16,185 images of 196 classes. Categories are typically at the level of Make, Model, Year, e.g., “2012 Tesla Model S” or “2012 BMW M3 coupe.” Compared to aircrafts, cars are smaller and appear in a more cluttered background. Thus object and part localization may play a more significant role here. This dataset was also part of the FGComp 2013 challenge.

NABirds [54] is larger than the CUB dataset consisting of 48,562 images of 555 species of birds that include most that are found in North America. The work engaged citizen scientists to produce high-quality annotations in a cost-effective manner. This dataset also provides parts and bounding-box annotations, but we only use category labels for training our models.

5.2.1 Bird species classification

Several methods report results requiring varying degrees of supervision such as part annotation or bounding-boxes at training and test time. We refer readers to [3] that has a comprehensive discussion of results on this dataset. A more up-to-date set of results can be found in [29] who recently reported excellent performance on this dataset leveraging more accurate CNN models with a method to train part detectors in a weakly supervised manner.

Comparison to baselines The fine-tuned FC-CNN [M] and FC-CNN [D] achieve accuracy of 58.8% and 70.4% respectively. Even without fine-tuning, the FV models achieve better results than the corresponding fine-tuned FC models, e.g., 61.1% using VGG-M and 71.3% using VGG-D networks. Surprisingly, VLAD models with CNN features outperforms corresponding FV models achieving 66.5% and 78% respectively. The NetVLAD approximation is as accurate as VLAD. With B-CNN models the results improve to 72.0% and 80.1%.

The results improve for all methods with fine-tuning. We evaluated the fine-tuned FV and VLAD models based on the fine-tuned FC models and found that this indirect fine-tuning improves performance, e.g., FV-CNN [D] improves to 74.7% and VLAD-CNN [D] improves to 79.0%. This shows that domain specific fine-tuning is useful even when early convolutional layers of a CNN are used. Direct fine-tuning of NetFV and NetVLAD leads to larger improvements, e.g., NetVLAD with VGG-D improves to 81.9% with fine-tuning. Even with indirect fine-tuning on FV models, we note that the FV-CNN results outperforms the multi-scale results reported in [9] – 49.9% using VGG-M and 66.7% using VGG-D. The B-CNN models are substantially more accurate than the corresponding FC, FV, VLAD models and their approximations.

The best model B-CNN [DM] achieves 84.1% accuracy on the CUB dataset. We additionally trained this model on the much larger NABirds dataset. We skip the SVM training and report the accuracy using softmax-layer predictions. We achieve 79.4% accuracy outperforming fine-tuned VGG-D by 15% (63.7%). The best performing method reported in the literature [54] achieves 75% using ground-truth part locations. Remarkably, our model significantly outperforms it without using and part or bounding-box annotations.

Comparison to previous work Two methods that perform well on this dataset when bounding-boxes are not available at test time are 73.9% of the “part-based R-CNN” [59] and 75.7% of the “pose-normalized CNN” [3]. Although the notion of parts differ, both these methods are based on a two step process of part detection followed by CNN based classifier. They also rely on part annotation during training. Our method outperforms these methods by a significant margin without relying on part or bounding-box annotations. Moreover, it is significantly simpler and
TABLE 2
Per-image accuracy on the birds [57], aircrafts [39] and cars [30] dataset for various methods described in Sec. 5.1. The first column lists the local features and the second column lists the encoding method. We consider SIFT, VGG-M network and VGG-D network for extracting local features.

For fully-connected (FC) models the features are extracted from the penultimate layer of the CNN, while for the texture representations the features are extracted from the relu5 and relu5_2 layer of VGG-M and VGG-D networks respectively. Results are shown without and with domain-specific fine-tuning. For VLAD and FV encoding we also report results using indirect fine-tuning (in gray italics) where features are constructed from the corresponding layers of the fine-tuned FC-CNN models. Direct fine-tuning using approximate models works significantly better. The first, second, and third highest-performing methods are marked with red, blue and yellow colors respectively.

| features   | encoding | birds      |                |                |                |                |                |
|------------|----------|------------|----------------|----------------|----------------|----------------|----------------|
|            |          | w/o ft     | w/ ft          | w/o ft         | w/ ft          | w/o ft         | w/ ft          |
| SIFT       | FV       | 18.8       | -              | 61.0           | -              | 59.2           | -              |
|            | FC       | 52.7       | 58.8           | 44.4           | 63.4           | 37.3           | 58.6           |
|            | NetBoVW  | 47.9       | 48.6           | 58.8           | 65.9           | 60.3           | 66.1           |
|            | VLAD     | 66.5       | 70.5           | 70.5           | 74.8           | 75.3           | 78.9           |
| VGG-M      | NetVLAD  | 66.8       | 72.1           | 70.7           | 76.7           | 76.0           | 83.7           |
|            | FV       | 61.1       | 64.1           | 64.3           | 71.2           | 70.8           | 77.2           |
|            | NetFV    | 64.5       | 71.7           | 68.6           | 75.5           | 72.3           | 81.8           |
|            | B-CNN    | 72.0       | 78.1           | 72.7           | 79.5           | 77.8           | 86.5           |
| VGG-D      | NetVLAD  | 77.9       | 81.9           | 75.3           | 81.8           | 82.1           | 88.6           |
|            | FV       | 71.3       | 74.7           | 70.4           | 78.7           | 75.2           | 85.7           |
|            | NetFV    | 73.9       | 79.9           | 71.5           | 79.0           | 77.9           | 86.2           |
|            | B-CNN    | 80.1       | 84.0           | 76.8           | 83.9           | 82.9           | 90.6           |
| VGG-M + VGG-D | B-CNN | 80.1 | 84.1 | 78.4 | 84.5 | 83.9 | 91.3 |
| Previous work |         | 84.1 [26], 82.0 [29] | 72.5 [6], 80.7 [23] | 92.6 [29], 82.7 [23] | 73.9 [59], 75.7 [3] | 78.0 [6] |

Faster. We note that the accuracy of these methods can be improved by replacing the underlying AlexNet CNN [31] with the more accurate but significantly slower VGG-D network. Krause et al. [29] reported 82.0% accuracy using a weakly supervised method to learn part detectors followed by the part-based analysis of [59] using the VGG-D network. However, this method relies on object bounding-boxes for training. Another recent approach called the “spatial transformer networks” reports 84.1% accuracy [26] using the Inception CNN architecture with batch normalization [25]. This approach also does not require object or part bounding-boxes at training time. Another recently proposed method that reports strong results on this setting is the “cross-layer pooling” method of [36] that considers pairwise features extracted from two different layers of a CNN. Using AlexNet they report an accuracy of 73.5%. Although not directly comparable, Krause et al. [28] show that by mining two orders of magnitude more labelled data by querying category labels on search engines, the performance of existing deep architectures can be improved to 92.1% on the CUB dataset. Such methods are complimentary to our approach and can be combined for additional benefits.

Common mistakes Fig. 4 shows the top six pairs of classes that are confused by our fine-tuned B-CNN [D,M] model. The most confused pair of classes is “American crow” and “Common raven”, which look remarkably similar. A quick search on the web reveals that the differences lie in the wing-spans, habitat, and voice, none of which are easy to measure from the image. Other commonly confused classes are also visually similar – various Shrikes, Terns, Flycatchers, Cormorants, etc. We note that the dataset has an estimated 4.4% label noise hence some of these errors may come from incorrect labeling [54].

5.2.2 Aircraft variant classification
Comparison to baselines The trends among the baselines are similar to those in birds with a few exceptions. The FV-SIFT baseline is remarkably good (61.0%) performing comparable to some of the fine-tuned FC-CNN baselines. Compared to the birds, the effect of fine-tuning FC-CNN [D] is significantly larger (45.0% → 76.6%) perhaps due to a larger domain shift from the ImageNet dataset.

The fine-tuned FV-CNN and VLAD-CNN models are also significantly better than the FC-CNN models in this dataset, where indirect fine-tuning via fine-tuning FC-CNN improves the models by 3-8%. Once again direct fine-tuning of approximate models leads to bigger improvements (e.g. VLAD 80.6% → NetVLAD 81.8% on VGG-D). The best performance of 84.5% is achieved by the B-CNN [D,M] model. Fine-tuning leads to 6% improvement in its accuracy.

Comparison to previous work This dataset does not come with part annotations hence several top performing methods for the birds dataset are not applicable here. We also compare against the results for “track
Fig. 2, i.e., w/o bounding-boxes, at the FGComp 2013 challenge website: https://sites.google.com/site/fgcomp2013/results. The best performing method [23] is a heavily engineered FV-SIFT which achieves 80.7% accuracy. Notable differences between our baseline FV-SIFT and theirs are (i) larger dictionary (256 → 1024), (ii) Spatial pyramid pooling (1×1 → 1×1 + 3×1), (iii) multiple SIFT variants, and (iv) multi-scale SIFT. The next best method is the “symbiotic segmentation” approach of [6] that achieves 72.5% accuracy. However, this method requires bounding-box annotations at training time to learn a detector which is refined to a foreground mask. The B-CNN models outperform these methods by a significant margin. The results on this dataset show that orderless pooling methods are still of considerable importance – they can be easily applied to new datasets as they only need image labels for training.

5.2.3 Car model classification

Comparison to baselines FV-SIFT once again does well on this dataset achieving 59.2% accuracy. Fine-tuning significantly improves performance of the FC-CNN models, e.g., 36.5% → 79.8% for FC-CNN [D], suggesting that the domain shift is larger here. The fine-tuned FC-CNN and VLAD-CNN models do significantly better, especially with the VGG-D network which obtains 85.7% and 85.6% accuracy respectively. With bilinear approximation, fine-tuning NetFV and NetVLAD improve the results to 86.2% and 88.6%. Once again the bilinear CNN models outperform all the other baselines with the B-CNN [D, M] model achieving 91.3% accuracy. Fine-tuning improves results by 7-8% for the B-CNN models.

Comparison to previous work The best accuracy on this dataset is 92.6% obtained by the recently proposed method [29]. We also compare against the winning methods from the FGComp 2013 challenge. The SIFT ensemble [23] won this category (during the challenge) achieving a remarkable 82.7% accuracy. The symbiotic segmentation achieved 78.0% accuracy. The fine-tuned B-CNN [D,M] obtains 91.3% significantly outperforming the SIFT ensemble, and nearly matching [29] which requires bounding-boxes during training. The results when bounding-boxes are available at test time can be seen in “track 1” of the FGComp 2013 challenge and are also summarized in [23]. The SIFT ensemble improves significantly with the addition of bounding-boxes (82.7% → 87.9%) in the cars dataset compared to aircraft dataset where it improves marginally (80.7% → 81.5%). This shows that localization in the cars dataset is more important than in aircrafts. Our bilinear models have a clear advantage over FV models in this setting since it can learn to ignore the background clutter.

5.3 Texture and scene recognition

We experiment on three texture datasets – the Describable Texture Dataset (DTD) [8], Flickr Material Dataset (FMD) [49], and KTH-TISP2-b (KTH-T2b) [4]. DTD consists of 5640 images labeled with 47 describable texture attributes. FMD consists of 10 material categories, each of which contains 100 images. Unlike DTD and FMD where images are collected from the Internet, KTH-T2b contains 4752 images of 11 materials captured under controlled scale, pose, and illumination. The KTH-T2b dataset splits the images into four samples for each category. We follow the standard protocol by training on one sample and test on the remaining three. On DTD and FMD, we randomly divide the dataset into 10 splits and report the mean accuracy across splits. Besides these, we also evaluate our models on MIT indoor scene dataset [44]. Indoor scenes are weakly structured and orderless texture representations have been shown to be effective here. The dataset consists of 67 indoor categories and a defined training and test split.

We compare B-CNN [D,D] to FV-CNN [D] without domain specific fine-tuning. Prior work [9] has shown that Fisher-vector representation with CNN filterbank responses achieve state-of-the-art results on various texture recognition benchmarks. Table 3 shows the results obtained by features extracted in single scale and by combining features from multiple scales 2s, s ∈ {1.5,0.5,3} relative to the 224×224 image using B-CNN and FV-CNN representations. We discard scales for which the image is smaller than the size of the receptive fields of the filters, or larger than 10242 pixels for efficiency. Across all scales of the input image the performance using B-CNN and FV-CNN is virtually identical. Multiple scales consistently lead to an improvement in accuracy. On the MIT indoor dataset fine-tuning the network using the B-CNN architecture leads to a small improvement 72.8% → 73.8% using relu5_3 and s = 1. The FV-CNN multi-scale results reported here are comparable (±1%) to the results reported in Cimpoi et al. [9] for all datasets except KTH-T2b (−4%). These differences in results are likely due to the choice of the CNN (they use the conv5_4 layer of the 19-layer VGG network) and the range of scales. These results show that the bilinear pooling is comparable to the Fisher-vector pooling for texture recognition. One drawback is that the FV features with 64 GMM components has half as many dimensions (64×2×256) as the bilinear features (256×256). However, it is known that these features are
Table 3

| Dataset  | FV-CNN  | B-CNN  |
|----------|---------|--------|
|          | \(s = 1\) | \(s = 2\) | \(m_s\) | \(s = 1\) | \(s = 2\) | \(m_s\) |
| DTD      | 67.8    | 70.6   | 73.6   | 69.6    | 71.5   | 72.9   |
|          | \(\pm 0.9\) | \(\pm 0.9\) | \(\pm 1.0\) | \(\pm 0.7\) | \(\pm 0.8\) | \(\pm 0.8\) |
| FMD      | 75.1    | 79.0   | 80.8   | 77.8    | 80.7   | 81.6   |
|          | \(\pm 2.3\) | \(\pm 1.4\) | \(\pm 1.7\) | \(\pm 1.9\) | \(\pm 1.5\) | \(\pm 1.7\) |
| KTH-T2b  | 74.8    | 75.9   | 77.9   | 75.1    | 76.4   | 77.9   |
|          | \(\pm 2.6\) | \(\pm 2.4\) | \(\pm 2.0\) | \(\pm 2.8\) | \(\pm 3.5\) | \(\pm 3.1\) |
| MIT indoor | 70.1    | 78.2   | 78.5   | 72.8    | 77.6   | 79.0   |

6 Analysis of Bilinear CNN Models

6.1 Dimensionality Reduction

The outer product between features generates very high dimensional image descriptors, e.g., 262K for the B-CNN models in Table 2, which might be undesirable for some applications. Our earlier work [34] showed that the features are highly redundant and their dimensionality can be reduced by an order of magnitude without loss in classification performance. Prior work [27] have also shown that in the context of SIFT-based Fisher vectors and VLAD highly compact representations can be obtained.

We investigate the tradeoff between accuracy and feature dimension for various models for fine-grained recognition. For NetVLAD and NetFV the feature dimension can be varied by changing the number of cluster centers. For B-CNN model, consider the case where the outer product is computed among features \(x\) and \(y\). There are several strategies for reducing the feature dimension:

1. Projecting the outer product into a lower dimensional space, i.e., \(\Phi(x, y) = \text{vec}(x^T y)P\). Here \(P\) is a projection matrix and \(\text{vec}\) operator reshapes the matrix into a vector.

2. Projecting both the features into a lower-dimensional space and computing outer product, \(\Phi(x, y) = \text{vec}(x\Lambda)^T(y\beta)\). Here \(\Lambda, \beta\) are projection matrices.

3. Projecting only one of the features into a lower-dimensional space and computing the outer product, i.e., \(\beta\) is an identity matrix.

In each case, the projection matrices can be initialized using principal component analysis (PCA). Although the first approach is straightforward, computing the PCA is computationally expensive due to the high dimensionality of the features (the covariance matrix of the outer product has \(d^4\) entries for \(d\)-dimensional features). The second approach is computationally attractive but the outer product of two PCA projected features results in a significant reduction in accuracy as shown in our earlier work [35], and more recently in [17]. Remarkably, we found that reducing the dimension of only one feature (third option) results in no significant loss in performance. While the projection can be initialized using PCA, they can be trained jointly with the classification layers. This technique was used in our earlier work [35] to reduce the feature dimension of B-CNN [M,M] models. It also breaks the symmetry of the features and is an example of a partially shared feature pipeline (Fig. 3(b)). It also resembles the computations of VLAD and Fisher vector representations where both \(f_A\) and \(f_B\) are based on the same underlying features.

Our experiments shown in Fig. 5 and Fig. 6 compare dimensionality reduction techniques for various models across datasets using the VGG-M model for both \(f_A\) and \(f_B\). The results are based on outputs of the softmax-layer, i.e., no follow-up SVM training is done. Fig. 5 shows a surprising trend for NetVLAD and NetFV representations. While increasing the number of cluster centers improves performance when the parameters are not fine-tuned in line with prior work [9], the performance decreases when fine-tuning is done in an end-to-end manner. This might reflect the increased difficulty in optimization when the number of components are increased.

Fig. 6 shows the performance of the asymmetric dimensionality reduction for the B-CNN model. We also compare these to a recently proposed “compact bilinear pooling” (CBP) [17] that approximates the outer product using a product of sparse linear projections of features with tensor sketches. The performance of the full model with \(512 \times 512\) dimensions is shown as the straight line in Fig. 6. On cars and aircraft datasets, dimensionality reduction by a factor of 64 even leads to a small improvement in accuracy. The projection initialized with PCA is slightly worse than CBP without fine-tuning, but is slightly better when the projections are fine-tuned jointly. Since the projection can be implemented as a matrix multiplication, empirically this is 1.5 times faster than CBP approach, which involves computing Fourier transforms and their inverses. Overall, for a given budget of dimensions the projected B-CNN models outperform NetVLAD and NetFV representations.

6.2 Training B-CNNs on ImageNet LSVRC

Here we experiment with training a B-CNN model from scratch on the ImageNet LSVRC 2012 dataset [47]. We experimenting with the effect of spatial jittering of training data on the classification performance. We train B-CNN [M,M] and VGG-M [7] with different amounts of spatial jittering – “f1” for flip, “f5” for flip + 5 translations and “f25” for flip + 25 translations. In each case the training is done using stochastic sampling where one of the jittered copies is randomly selected for each example. The network parameters are randomly initialized and trained using stochastic gradient descent with momentum for a number of epochs. We start with a high learning rate and reduce it by a factor of 10 when the validation error stops decreasing. We stop training when the validation error stops decreasing.
Table 4 shows the “top1” and “top5” validation errors and compares the B-CNN network to the standard VGG-M model. The validation error is reported on a single center cropped image. Note that we train all networks with neither PCA color jittering nor batch normalization and our baseline results are within 2% of the top1 errors reported in [7]. The VGG-M model achieves 46.4% top1 error with flip augmentation during training. The performance improves significantly to 39.6% with f25 augmentation. As fully connected layers in a standard CNN network encode spatial information, the model loses performance without spatial jittering. For B-CNN network, the model achieves 38.7% top1 error with f1 augmentation, outperforming VGG-M with f25 augmentation. With more augmentations, B-CNN model improves top1 error by 1.6% (38.7% → 37.1%). Going from f5 to f25, B-CNN model improves marginally by < 1%. The results show that B-CNN feature is discriminative and robust to translation. With a small amount of data jittering, B-CNN network achieves fairly good performance, suggesting that explicit translation invariance might be preferable to the implicit invariance obtained by data jittering.

### Table 4

| data aug. | B-CNN [M,M] | VGG-M |
|-----------|-------------|-------|
|           | f1 | f5 | f25 | f1 | f25 |
| error@1   | 38.7 | 37.1 | 36.6 | 46.4 | 39.6 |
| error@5   | 17.0 | 16.3 | 16.0 | 22.5 | 17.6 |

#### 6.3 Visualizing CNN filters

One of the motivations for the bilinear model was the modular separation of factors that affect the overall appearance. But do the networks specialize into roles of localization (“where”) and appearance modeling (“what”) when initial-
and normalizing the features (signed square-root and $\ell_2$). Let $r_i, i = 1, \ldots, n$, be the index of the $i^{th}$ layer with the bilinear feature representation $B_{r_i}$ from which we obtain category prediction probabilities $C_{r_i}$ by training a linear classifier in a supervised manner. Given a target category $C$ we can obtain an image that matches the target label by solving the following optimization:

$$\min_x \sum_{i=1}^{m} L \left( C_{r_i}, \hat{C} \right) + \gamma \Gamma(x).$$

Here, $L$ is a loss function such as the negative log-likelihood of the label $\hat{C}$ and $\gamma$ is a tradeoff parameter. The image prior $\Gamma(x)$ encourage the smoothness of output image where we use the TV$_\beta$ norm with $\beta = 2$:

$$\Gamma(x) = \sum_{i,j} \left( (x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2 \right)^{\beta/2}.$$  

The exponent $\beta = 2$ was empirically found to lead to better reconstructions in [38] as it leads to fewer “spike” artifacts than $\beta = 1$. We refer readers to our earlier work [34] where this framework was applied to texture synthesis, blending texture attributes, and image manipulation with texture attributes.

Visualizing texture categories We use the 16-layer VGG network [51] pre-trained on ImageNet for inverting categories. In our experiments, given an input image we resize it to $224 \times 224$ pixels before computing the target bilinear features and solve for $x \in \mathbb{R}^{224 \times 224 \times 3}$. This is primarily for speed since the dimension of the bilinear features are independent of the size of the image. We optimize the log-likelihood of the class probability using classifiers trained on bilinear features at $\text{relu}_2$, $\text{relu}_3$, $\text{relu}_4$, $\text{relu}_5$. We use L-BFGS for optimization and compute the gradients of the objective with respect to $x$ using back-propagation. By choosing different mapping functions $g_A$ and $g_B$, we show the inverse images for various categories from various texture models.

Fig. 8 shows some example inverse images for various categories for the DTD, FMD and MIT indoor datasets using B-CNN. These images were obtained by setting $\gamma = 1 \epsilon - 8$, and $\hat{C}$ to various class labels in Eqn. 5. These images reveal how the model represents texture and scene categories. For instance, the dotted category of DTD contains images of various colors and dot sizes and the inverse image is composed of multi-scale multi-colored dots. The inverse images of water and wood from FMD are highly representative of these categories. Note that these images cannot be obtained by simply averaging instances within a category which is likely to produce a blurry image. The orderless nature of the texture descriptor is essential to produce such sharp images. The inverse scene images from the MIT indoor dataset reveal key properties that the model learns – a bookstore is visualized as racks of books while a laundromat has laundry machines at various scales and locations. In Fig. 9 we visualize reconstructions by incrementally adding layers in the texture representation. Lower layers preserve color and small-scale structure and combining all the layers leads to better reconstructions. Even though the $\text{relu}_5$ layer provides the best recognition accuracy, simply using that layer did not produce good inverse images (not shown).
7 Conclusion

We present bilinear CNN models which generalize classical texture representations and relate them to the existing literature of part-based models and attention-based models. Bilinear models can be trained end-to-end and combined with existing deep architectures in a seamless manner. We also compare the proposed models with explicit approximations of various texture features and study the accuracy and memory trade-offs they offer. The main conclusion is that variants of outer-product representations are very effective at various fine-trained, texture, and scene recognition tasks. Moreover, these representations are highly redundant and in some cases their dimension can be reduced by two orders of magnitude without significant loss in accuracy. We also present a study of how these models perform when trained from scratch on ImageNet classification task, as well as visualizations of the learned models using top activations of various filters and category inverses.

Acknowledgement

This research was supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) under contract number 2014-14071600010, National Science Foundation (NSF) grant IIS-1617917, and a Faculty gift from Facebook. The GPUs used in this research were generously donated by NVIDIA.

References

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 2, 3, 5
[2] L. Bourdev, S. Maji, and J. Malik. Describing people: A poselet-based approach to attribute classification. In The IEEE International Conference on Computer Vision (ICCV), 2011. 4
[3] S. Branson, G. V. Horn, S. Belongie, and P. Perona. Bird species categorization using pose normalized deep convolutional nets. In British Machine Vision Conference (BMVC), 2014. 4, 6, 7
[4] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. In The IEEE International Conference on Computer Vision (ICCV), 2005. 8
[5] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In European Conference on Computer Vision (ECCV). 2012. 5

[6] Y. Chai, V. Lempitsky, and A. Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In The IEEE International Conference on Computer Vision (ICCV), 2013. 7, 8

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In British Machine Vision Conference (BMVC). 2014. 9, 10

[8] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. 2, 4, 5, 8, 12

[9] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi. Deep filter banks for texture recognition, description, and segmentation. International Journal of Computer Vision, 118(1):65–94, 2016. 1, 4, 5, 6, 8, 9

[10] G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In ECCV Workshop on Statistical Learning in Computer Vision, 2014. 3

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 886–893, 2005. 1

[12] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In International Conference on Machine Learning, 2013. 2, 4

[13] R. Farrell, O. Oza, N. Zhang, V. I. Morariu, T. Darrell, and L. S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In The IEEE International Conference on Computer Vision (ICCV), 2011. 4

[14] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 5

[15] K. Fragiadaki, P. Arbelaez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. 5

[16] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. CoRR, abs/1606.01847, 2016. 5

[17] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. 2, 9, 10

[18] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. In Advances in Neural Information Processing Systems, 2015. 4

[19] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. 4

[20] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In Advances in Neural Information Processing Systems, pages 3248–3256, 2012.

[21] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. 2

[22] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In European Conference on Computer Vision (ECCV), 2014. 1

[23] P.-H. Gosselin, N. Murray, H. Jegou, and F. Perronnin. Revisiting the fisher vector for fine-grained classification. Pattern Recognition Letters, 49(92–98):2014. 4, 7, 8

[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. 4

[25] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, 2015. 7

[26] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu. Spatial transformer networks. In Advances in Neural Information Processing Systems, pages 2017–2025, 2015. 4, 7

[27] H. Jegou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010. 1, 3, 9

[28] A. H. H. Z. A. T. D. J. P. L. F.-F. Jonathan Krause, Benjamin Sapp. The unreasonable effectiveness of noisy data for fine-grained recognition. 2016. 7

[29] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. 6, 7, 8

[30] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. Object representations for fine-grained categorization. In 3D Representation and Recognition Workshop, at ICCV, 2013. 6, 7

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, 2012. 1, 4, 7

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceddings of the IEEE, 86(11):2278–2324, 1998. 1

[33] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textures. International journal of Computer Vision, 43(1):29-44, 2001. 4

[34] T.-Y. Lin and S. Maji. Visualizing and understanding deep texture representations. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. 2, 9, 11

[35] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In The IEEE International Conference on Computer Vision (ICCV), 2015. 2, 5, 9, 11, 12

[36] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. 7

[37] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European Conference on Computer Vision (ECCV), 2016. 5

[38] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. International Journal of Computer Vision (IJCV), 2016. 2, 11

[39] S. Maji, E. Rabah, L. L. M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. arXiv preprint
[40] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[41] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007.

[42] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In European Conference on Computer Vision (ECCV), 2010.

[43] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In Advances in Neural Information Processing Systems, 2009.

[44] A. Quattoni and A. Torralba. Recognizing indoor scenes. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[45] A. S. Razavin, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In DeepVision workshop, 2014.

[46] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99. 2015.

[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015.

[48] B. Scholkopf and A. J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2001.

[49] L. Sharan, R. Rosenholz, and E. H. Adelson. Material perception: What can you see in a brief glance? Journal of Vision, 9:784(8), 2009.

[50] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Advances in Neural Information Processing Systems, 2014.

[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations, 2015.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

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

[54] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[55] A. Vedaldi and B. Fulkerson. VLFeat: an open and portable library of computer vision algorithms. In ACM International Conference on Multimedia, 2010.

[56] A. Vedaldi and K. Lenc. MatConvNet – Convolutional Neural Networks for MATLAB. In ACM International Conference on Multimedia, 2015.

[57] G. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, CalTech, 2011.

[58] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In European Conference on Computer Vision, pages 818–833. Springer, 2014.

[59] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based R-CNNs for fine-grained category detection. In European Conference on Computer Vision (ECCV), 2014.

[60] N. Zhang, R. Farrell, and T. Darrell. Pose pooling kernels for subcategory recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.