Bilinear CNNs for Fine-grained Visual Recognition

Tsung-Yu Lin Aruni RoyChowdhury Subhransu Maji

Abstract—We present a simple and effective architecture for fine-grained visual recognition called *Bilinear Convolutional Neural Networks (B-CNNs)*. These networks represent an image as a *pooled outer product* of features derived from two CNNs and capture localized feature interactions in a translationally invariant manner. B-CNNs belong to the class of orderless texture representations but unlike prior work they can be trained in an end-to-end manner. Our most accurate model obtains 84.1%, 79.4%, 86.9% and 91.3% per-image accuracy on the Caltech-UCSD birds [67], NABirds [64], FGVC aircraft [42], and Stanford cars [33] dataset respectively and runs at 30 frames-per-second on a NVIDIA Titan X GPU. We then present a systematic analysis of these networks and show that (1) the bilinear features are highly redundant and can be reduced by an order of magnitude in size without significant loss in accuracy, (2) are also effective for other image classification tasks such as texture and scene recognition, and (3) can be trained from scratch on the ImageNet dataset offering consistent improvements over the baseline architecture. Finally, we present visualizations of these models on various datasets using top activations of neural units and gradient-based inversion techniques. The source code for the complete system is available at http://vis-www.cs.umass.edu/bcnn.

Index Terms—Fine-grained recognition, Texture representations, Second-order pooling, Bilinear models, Convolutional networks

1 Introduction

Fine-grained recognition involves classification of instances within a subordinate category. Examples include recognition of species of birds, models of cars, or breeds of dogs. These tasks often require recognition of highly localized attributes of objects while being invariant to their pose and location in the image. For example, distinguishing a "California gull" from a "Ringed-bill gull" requires the recognition of patterns on their bill, or subtle color differences of their feathers [1]. There are two broad classes of techniques that are effective for these tasks. *Part-based models* construct representations by localizing parts and extracting features conditioned on their detected locations. This makes subsequent reasoning about appearance easier since the 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 classical image representations, such as Bag-of-Visual-Words [12] and their variants popularized for texture analysis. Most modern approaches are based on representations extracted using Convolutional Neural Networks (CNNs) pre-trained on the ImageNet dataset [54]. While part-based models based on CNNs are more accurate, they require part annotations during training. This makes them less applicable in domains where such annotations are difficult or expensive to obtain, including categories without a clearly defined set of parts such as textures and scenes.

In this paper we argue that the effectiveness of part-based reasoning is due to their invariance to position and pose of the object. Texture representations are translationally invariant by design as they are based on aggregation of local image features in an *orderless* manner. While classical texture representations based on SIFT [40] and their recent extensions based on CNNs [11], [24], have been shown to be effective at fine-grained recognition, they have not matched the performance of part-based approaches. A potential reason for this gap is that the underlying features in texture representations are not learned in an end-to-end manner and are likely to be suboptimal for the recognition task.

We present *Bilinear CNNs (B-CNNs)* that address several drawbacks of existing deep texture representations. Our key insight is that several widely-used texture representations can be written as a pooled outer product of two suitably designed features. When these features are based on CNNs the resulting architecture consists of standard CNN units for feature extraction, followed by a specially designed *bilinear layer* and a *pooling layer*. The output is a fixed high-dimensional representation which can be combined with a fully-connected layer to predict class labels. The simplest bilinear layer is one where two *identical* features are combined with an outer product. This is closely related to the *Second-Order Pooling* approach of Carreira *et al*. [8] popularized for semantic image segmentation. We also show that other texture representations can be written as B-CNNs once suitable non-linearities are applied to the underlying features. This results in a family of layers which can be plugged into existing CNNs for end-to-end training on large datasets, or domain-specific fine-tuning for transfer learning. B-CNNs outperform existing models, including those trained with part-level supervision, on a variety of fine-grained recognition datasets. Moreover, these models are fairly efficient. Our most accurate model implemented in MatConvNet [66] runs at 30 frames-per-second on a NVIDIA Titan X GPU and obtains **84.1%**, **79.4%**, **86.9%** and **91.3%** per-image accuracy on Caltech-UCSD birds [67], NABirds [64], FGVC aircraft [42], and Stanford cars [33] dataset respectively.

- *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*

This manuscript combines the analysis of our earlier works [36], [37] and extends them in a number of ways. We present an account of related work, including extensions published subsequently (Section 2). We describe the B-CNN architecture (Section 3), and present a unified analysis of exact and approximate end-to-end trainable formulations of Second-Order Pooling (O2P), Fisher Vector (FV), Vector-of-Locally-Aggregated Descriptors (VLAD), Bag-of-Visual-Words (BoVW) in terms of their accuracy on a variety of fine-grained recognition datasets (Section 3.2-4). We show that the approach is general-purpose and is effective at other image classification tasks such as material, texture, and scene recognition (Section 4). We present a detailed analysis of dimensionality reduction techniques and provide trade-off curves between accuracy and dimensionality for different models, including a direct comparison with the recently proposed *compact bilinear pooling* technique [19] (Section 5.1). Moreover, unlike prior texture representations based on networks pre-trained on the ImageNet dataset, B-CNNs can be trained from scratch and offer consistent improvements over the baseline architecture with a modest increase in the computation cost (Section 5.2). Finally we visualize the top activations of several units in the learned models and apply the gradient-based technique of Mahendran and Vedaldi [41] to visualize inverse images on various texture and scene datasets (Section 5.3). We have released the complete source code for the system at http://vis-www.cs.umass.edu/bcnn.

2 Related work

Fine-grained recognition techniques. After AlexNet's [34] impressive performance on the ImageNet classification challenge, several authors (*e.g.*, [13], [52]) have demonstrated that features extracted from layers of a CNN are effective at fine-grained recognition tasks. Building on prior work on part-based techniques (*e.g.*, [5], [15], [71]), Zhang *et al*. [70], and Branson *et al*. [6] demonstrated the benefits of combining CNN-based part detectors [23] and CNN-based features for fine-grained recognition tasks. Other approaches use segmentation to guide part discovery in a weakly-supervised manner and train part-based models [31]. Among the non part-based techniques, texture descriptors such as FV and VLAD have traditionally been effective for fine-grained recognition. For example, the top performing method on FGCOMP'12 challenge used SIFT-based FV representation [25].

Recent improvements in deep architectures have also resulted in improvements in fine-grained recognition. These include architectures that have increased depth such as the "deep" [9] and "very deep" [59] networks from the Oxford's VGG group, inception networks [60], and "ultra deep" residual networks [26]. Spatial Transformer Networks [29] augment CNNs with parameterized image transformations and are highly effective at fine-grained recognition tasks. Other techniques augment CNNs with "attention" mechanisms that allow focused reasoning on regions of an image [4], [43]. B-CNNs can be viewed as an implicit spatial attention model since the outer product modulates one feature based on the other, similar to the multiplicative feature interactions in attention mechanisms. Although not directly comparable, Krause *et al*. [32] showed that the accuracy of deep networks can be improved significantly by using two orders of magnitude more training data obtained by querying category labels on search engines. Recently, Moghimi *et al*. [44] showed boosting B-CNNs offers consistent improvements on fine-grained tasks.

Texture representations and second-order features. Texture representations have been widely studied for decades. Early work [35] represents the texture by computing the statistics of linear filter-bank responses (*e.g.*, wavelets and steerable pyramids). The use of second-order features of filter-bank responses was pioneered by Portilla and Simoncelli [50]. Recent variants such as FV [46] and O2P [8] with SIFT were shown to be a highly effective for image classification and semantic segmentation [14] tasks respectively.

The advantages of combining orderless texture representations and deep features have been studied in a number of recent works. Gong *et al*. performed a *multi-scale orderless pooling* of CNN features [24] for scene classification. Cimpoi *et al*. [11] performed a systematic analysis of texture representations by replacing linear filter-banks with non-linear filter-banks derived from a CNN and showed it results in significant improvements on various texture, scene, and fine-grained recognition tasks. They found that orderless aggregation of CNN features was more effective than the commonly-used fully-connected layers on these tasks. However, a drawback of these approaches is that the filter banks are not trained in an end-to-end manner. Our work is also related to the *cross-layer pooling* approach of Liu *et al*. [38] who showed that second-order aggregation of features from two different layers of a CNN is effective at fine-grained recognition. Our work showed that feature normalization and domain-specific fine-tuning offers additional benefits, improving the accuracy from 77.0% to 84.1% using identical networks on the Caltech-UCSD Birds dataset [67]. Another subsequently published work of interest is the *NetVLAD* architecture [3] which provides a end-to-end trainable approximation of VLAD. The approach was applied to image-based geolocation problem. We include a comparison of NetVLAD to other texture representations in Section 4.

Texture synthesis and style transfer. Concurrent to our work, Gatys *et al*. showed that the Gram matrix of CNN features is an effective texture representation and by matching the Gram matrix of a target image one can create novel images with the same texture [20] and transfer styles [21]. While the Gram matrix is identical to a pooled bilinear representation when the two features are the same, the emphasis of our work is *recognition* and *not generation*. This distinction is important since Ustyuzhaninov *et al*. [63] show that the Gram matrix of a shallow CNN with random filters is sufficient for texture synthesis, while discriminative pre-training and subsequent fine-tuning are essential to achieve high performance for recognition.

Polynomial kernels and sum-product networks. An alternate strategy for combining 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 [55]. However, one might be able to achieve similar approximations using alternate architectures such as sum-product networks that efficiently model multiplicative interactions [22].

Bilinear model variants and extensions. Bilinear models were used by Tanenbaum and Freeman [62] 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 [49] that express the classifier as a product of two low-rank matrices. Our models based on low dimensional representations described in Section 5.1 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 [17], [58]. 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 [18] where text and visual features are combined, action recognition [16] where optical flow and image features are combined.

Low-dimensional bilinear features. A drawback of the bilinear features is the memory overhead of storing the high-dimensional features. For example, the outer product of 512 dimensional features results in a 512×512 dimensional representation. Our earlier work [37] showed that the overall representation can be reduced to 512×64 dimensions by projecting one of the features to a lower-dimensional space. Alternatively, the *compact bilinear pooling* [19] applies *tensor sketching* [48] to aggregate low-dimensional embeddings that approximate the bilinear features. In Section 5.1 we compare the two approaches and find that the projection method is simpler, faster, and equally effective. In most cases features size can be reduced 8-32× without significant loss in accuracy.

Scalability and speed. B-CNNs compare favorably to traditional CNN architectures in terms of speed since they replace several fully-connected layers with a bilinear pooling layer and a linear layer. Our MatConvNet-based [66] 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 [53] or Single-Shot Detector (SSD) [39], part-based models for fine-grained recognition are 2-10× slower. The main advantage of B-CNNs is that they require image labels only and can be easily applied to different fine-grained datasets.

3 B-CNNs for Image Classification

In this section we introduce the B-CNN architecture for image classification and then show that various widely used texture representations can be written as B-CNNs.

3.1 The B-CNN architecture

A B-CNN for image classification consists of a quadruple $\mathcal{B} = (f_A, f_B, \mathcal{P}, \mathcal{C})$. Here f_A and f_B are *feature functions* based on CNNs, $\mathcal{P}$ is a *pooling function*, and $\mathcal{C}$ is

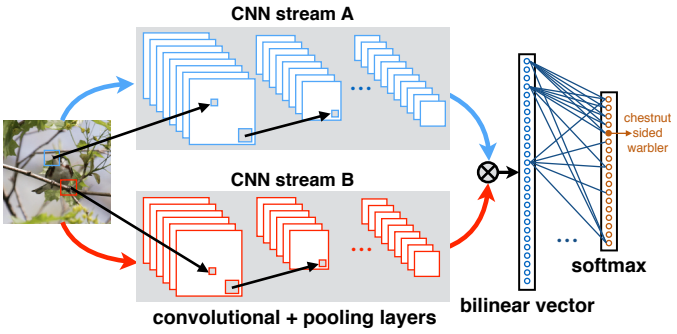


Fig. 1. **Image classification using a B-CNN.** An image is passed through CNNs A and B, and their outputs at each location are combined using the matrix outer product and average pooled to obtain the bilinear feature representation. This is passed through a linear and softmax layer to obtain class predictions.

a *classification function*. A feature function is a mapping $f : \mathcal{L} \times \mathcal{I} \rightarrow \mathbb{R}^{K \times D}$, that takes an image $I \in \mathcal{I}$ and a location $l \in \mathcal{L}$ and outputs a feature of size $K \times D$. 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 combination of f_A and f_B at a location l is given by

$$\text{bilinear}(l, I, f_A, f_B) = f_A(l, I)^T f_B(l, I). \tag{1}$$

Both f_A and f_B must have the same feature dimension K to be compatible. The value of K depends on the particular model. For example, $K = 1$ for BoVW model and equals the number of clusters in a FV model (details in Section 2). The pooling function $\mathcal{P}$ aggregates the bilinear combination of features across all locations in the image to obtain a global image representation $\Phi(I)$. We use sum pooling in all our experiments, *i.e.*,

$$\Phi(I) = \sum_{l \in \mathcal{L}} \text{bilinear}(l, I, f_A, f_B) = \sum_{l \in \mathcal{L}} f_A(l, I)^T f_B(l, I). \tag{2}$$

Since the location of features is ignored during pooling, the bilinear feature $\Phi(I)$ is an *orderless* representation. If f_A and f_B extract features of size $K \times M$ and $K \times N$ respectively, then $\Phi(I)$ is of size $M \times N$. The bilinear feature is a general-purpose image representation that can be used with a classifier $\mathcal{C}$ (Figure 1). Intuitively, the outer product conditions the outputs of features f_A and f_B on each other by considering their pairwise interactions, similar to the feature expansion in a quadratic kernel.

3.1.1 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 *truncated* at an intermediate layer as feature functions. By pre-training we benefit when domain-specific data is limited. This has been shown to be effective for a number of tasks ranging from object detection, texture recognition, to fine-grained classification [10], [13], [23], [52]. Another advantage of using CNNs is that the resulting

network can process images of an arbitrary size and produce outputs indexed by image location and feature channel.

Our earlier work [37] experimented with models where the feature functions f_A and f_B were either *independent* or *fully shared*. Here we also experiment with feature functions that share a part of the feed-forward computation as seen in Figure 3. The feature functions used to approximate classical texture representations we present in Section 3.2, as well as the low-dimensional B-CNNs we present in Section 5.1.

3.1.2 Normalization and classification

We perform additional normalization steps where the bilinear feature $\mathbf{x} = \Phi(I)$ is passed through a signed square-root ($\mathbf{y} \leftarrow \text{sign}(\mathbf{x})\sqrt{|\mathbf{x}|}$), followed by ℓ_2 normalization ($\mathbf{z} \leftarrow \mathbf{y}/||\mathbf{y}||_2$) inspired by [47]. This improves performance in practice (see our earlier work [37] for an evaluation of the effect of normalization). For classification we use logistic regression or linear SVM [55]. Although this can be replaced with an arbitrary multi-layer network, we found that linear models are effective on top of bilinear features.

3.1.3 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.*, cross-entropy). The *bilinear form* simplifies the gradient computations. If the outputs of the two networks are matrices A and B of size $L \times M$ and $L \times N$ respectively, then the bilinear feature is $\mathbf{x} = A^T B$ of size $M \times N$. Let $d\ell/d\mathbf{x}$ be the gradient of the loss function ℓ with respect to $\mathbf{x}$, then by chain rule of gradients we have:

$$\frac{d\ell}{dA} = B\left(\frac{d\ell}{d\mathbf{x}}\right)^T, \quad \frac{d\ell}{dB} = A\left(\frac{d\ell}{d\mathbf{x}}\right). \tag{3}$$

As long as the gradients of the features A and B can be computed efficiently the entire model can be trained in an end-to-end manner. The scheme is illustrated in Figure 2.

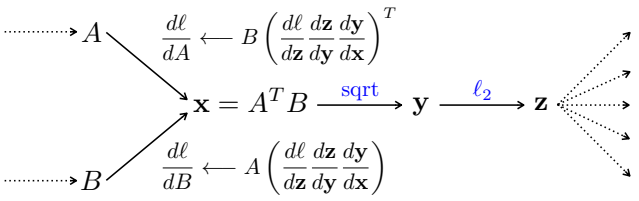


Fig. 2. Flow of gradients in a B-CNN.

3.2 Relation to classical texture representations

In this section we show that various *orderless* texture descriptors can be written in the bilinear form and derive variants that are end-to-end trainable. Since the properties of texture are usually translationally invariant, most texture representations are based on orderless aggregation of local image features, *e.g.*, sum or max operation. A non-linear encoding is typically applied before aggregation of local features to improve their representation power. Additionally, a normalization of the aggregated feature (*e.g.*, power and ℓ_2) is done to increase invariance. Thus, texture representations can be defined by the choice of the *local features*, the *encoding function*, the *pooling function*, and the *normalization function*. To simplify further analysis, we will decompose the feature function f as $f(l, \mathcal{I}) = g(h(l, \mathcal{I})) = g(\mathbf{x})$ to denote the explicit dependency on the image and the location of h and additional non-linearities g.

One of the earliest representation used for texture is the Bag-of-Visual-Words (BoVW) [12]. It was shown to be effective at several recognition tasks beyond texture. While variants differ on how the visual words are learned, a popular approach is to obtain a set of k centers by clustering image features (*e.g.*, using k-means). Each feature $\mathbf{x}$ is then assigned to the closest cluster center (also called "hard assignment") and the image is represented as a histogram denoting frequencies of each visual word. If we denote $\eta(\mathbf{x})$ as the *one-hot encoding* that is 1 at the index of the closest center of $\mathbf{x}$ and zero elsewhere, then BoVW can be written as a bilinear model with $g_A(\mathbf{x}) = 1$ and $g_B(\mathbf{x}) = \eta(\mathbf{x})$.

The VLAD representation [30] encodes a descriptor $\mathbf{x}$ as $(\mathbf{x} - \mu_k) \otimes \eta(\mathbf{x})$, where $\otimes$ is the *kronecker product*, μ_k is the closest center to $\mathbf{x}$, and $\eta(\mathbf{x})$ is the one-hot encoding of $\mathbf{x}$ as before. These encodings are aggregated across the image by sum pooling. Thus VLAD can be written as a bilinear model with $g_A(\mathbf{x}) = [\mathbf{x} - \mu_1; \mathbf{x} - \mu_2; \ldots; \mathbf{x} - \mu_k]$. Here, g_A has k rows each corresponding to a center. And $g_B(\mathbf{x}) = \text{diag}(\eta(\mathbf{x}))$, a matrix with $\eta(\mathbf{x})$ in the diagonal and 0 elsewhere. Notice that the feature functions for VLAD output a matrix with $k > 1$ rows at each location.

The FV representation [47] computes both the first order $\alpha_i = \Sigma_i^{-\frac{1}{2}}(\mathbf{x} - \mu_i)$ and second order $\beta_i = \Sigma_i^{-1}(\mathbf{x} - \mu_i) \odot (\mathbf{x} - \mu_i) - 1$ statistics, which are aggregated and weighted by the Gaussian mixture model (GMM) posteriors $\theta(\mathbf{x})$. Here μ_i and Σ_i are the mean and covariance of the i^{th} GMM component respectively and $\odot$ denotes element-wise multiplication. Thus, FV can be written as a bilinear model with $g_A = [\alpha_1\ \beta_1; \alpha_2\ \beta_2; \ldots; \alpha_k\ \beta_k]$ and $g_B = \text{diag}(\theta(\mathbf{x}))$.

The O2P representation [8] computes the covariance statistics of SIFT features within a region, followed by log-Euclidean mapping and power normalization. Their approach was shown to be effective for semantic segmentation. O2P can can be written as a bilinear model with symmetric features, *i.e.*, $f_A = f_B = f_{\text{sift}}$, followed by pooling and non-linearities.

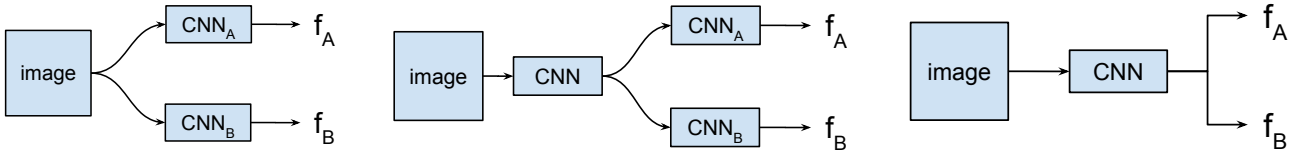
The appearance-based cluster centers learned by the encoder, $\eta(\mathbf{x})$ or $\theta(\mathbf{x})$, in the BoVW, VLAD and FV representations can be thought of as part detectors. Indeed it has been observed that the cluster centers tend to localize facial landmarks when trained on faces [45]. Thus, by modeling the joint statistics of the encoder $\eta(\mathbf{x})$ or $\theta(\mathbf{x})$, and the appearance $\mathbf{x}$, 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.

3.2.1 End-to-end trainable formulations

Prior work on fine-grained recognition using texture encoders [11], [25] did not learn the features in an end-to-end manner. Below we describe a recently proposed end-to-end trainable approximation of VLAD, and present similar formulations for all texture representations described in the earlier section. The ability to directly fine-tune these

TABLE 1
Texture encoders such as VLAD, FV, BoVW and O2P can be written as outer products of the form ${g_A}^T g_B$. On the right are their end-to-end trainable formulations that simplify gradient computations by replacing "hard assignment" η with "soft assignment" $\bar{\eta}$, ignoring variance normalization for FV, *etc.* For the symmetric case (*i.e.*, when $f_A = f_B$) bilinear pooling is identical to O2P. See Section 3.2.1 for details.

	Exact formulation		End-to-end trainable formulation	
Model	$g_B(\mathbf{x})$	$g_A(\mathbf{x})$	$g_B(\mathbf{x})$	$g_A(\mathbf{x})$
VLAD	$\text{diag}(\eta(\mathbf{x}))$	$\mathbf{x}-\boldsymbol{\mu}$	$\text{diag}(\bar{\eta}(\mathbf{x}))$	$\mathbf{x}-\boldsymbol{\mu}$
FV	$\text{diag}(\theta(\mathbf{x}))$	$[\boldsymbol{\alpha}, \boldsymbol{\beta}]$	$\text{diag}(\bar{\eta}(\mathbf{x}))$	$[\mathbf{x}-\boldsymbol{\mu}, (\mathbf{x}-\boldsymbol{\mu}) \odot (\mathbf{x}-\boldsymbol{\mu})]$
BoVW	$\eta(\mathbf{x})$	1	$\bar{\eta}(\mathbf{x})$	1
O2P	$\mathbf{x}$	$\mathbf{x}$	$\mathbf{x}$	$\mathbf{x}$



(a) no sharing (b) partially shared (c) fully shared

Fig. 3. Feature functions in B-CNNs can (a) share no computations (*e.g.*, B-CNN model based on VGG-M and VGG-D), (b) share computations partially (*e.g.*, NetVLAD, B-CNN PCA model described in Section 5.1), and (c) share all computations (*e.g.*, B-CNN model based on VGG-M).

models leads to significant improvements in accuracy across a variety of fine-grained datasets.

Recently, an end-to-end trainable formulation of VLAD called NetVLAD was proposed by Arandjelović *et al*. [3]. The first simplification was to replace the "hard assignment" $\eta(\mathbf{x})$ in g_B by a differentiable "soft assignment" $\bar{\eta}(\mathbf{x})$. Given the k-th cluster center μ_k, the k-th component of the soft assignment vector for an input $\mathbf{x}$ is given by,

$$\bar{\eta}_k(\mathbf{x}) = \frac{e^{-\gamma||\mathbf{x}-\mu_k||^2}}{\sum_{k'} e^{-\gamma||\mathbf{x}-\mu_{k'}||^2}} = \frac{e^{{\mathbf{w}_k}^T\mathbf{x}+b_k}}{\sum_{k'} e^{{\mathbf{w}_{k'}}^T\mathbf{x}+b_{k'}}} \quad (4)$$

where $\mathbf{w}_k = 2\gamma\mu_k$, $b_k = -\gamma||\mu_k||^2$ and γ 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 $[\mathbf{x}-\mu_1; \mathbf{x}-\mu_2; \ldots; \mathbf{x}-\mu_k]$. The second simplification is to decouple the dependence on μ of both the g_A and g_B during training which makes gradient computation easier. Thus in NetVLAD during training the weights $\mathbf{w}_k$, b_k and μ_k are independent parameters.

We extend the NetVLAD to NetFV by appending the second order statistics to the feature g_A, *i.e.*, $g_A = [\mathbf{x}-\mu_1, (\mathbf{x}-\mu_1)^2; \mathbf{x}-\mu_2, (\mathbf{x}-\mu_2)^2; \ldots; \mathbf{x}-\mu_k, (\mathbf{x}-\mu_k)^2]$. Here, the squaring is done in an element-wise manner, *i.e.*, $(\mathbf{x}-\mu_i)^2 = (\mathbf{x}-\mu_i) \odot (\mathbf{x}-\mu_i)$. The feature g_B is kept identical to NetVLAD. This simplification discards the covariances and priors present in the true GMM posterior used in the FV model. Similarly, the *NetBoVW* approximation to BoVW replaces the hard assignments by soft assignments $\bar{\eta}(\mathbf{x})$ computed in a manner similar to NetVLAD.

The O2P representation is identical to B-CNN when the feature functions f_A and f_B are identical. However, the O2P representation applies a log-Euclidean (matrix-logarithm) mapping to the pooled representation which is rather expensive to compute since it involves an Eigenvalue decomposition and currently does not have efficient implementation on GPUs. This significantly slows the forward and gradient computations of the entire network. Skipping this step allows us to efficiently fine-tune the model. We also note that concurrent to our publication [37], Ionescu *et al*. [28] proposed a DeepO2P approach and noted similar difficulties.

4 Image Classification Experiments

We outline the models used in our experiments in Section 4.1. We then provide a comparison of various B-CNNs to prior work on fine-grained recognition in Section 4.2, and texture and scene recognition in Section 4.3.

4.1 Models

Below we describe various models used in our experiments:

FV with SIFT. We implemented a FV based using dense SIFT features [47] extracted using VLFEAT [65]. The image is first resized to 448×448 and SIFT features with a *bin size* 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 (FC) layers. This is a standard baseline where the features are extracted from the last FC layer, *i.e.*, before the `softmax` layer of a CNN. The input image is resized to 224×224 (the input size of the CNN) and mean-subtracted before propagating it though the CNN. We consider two different representations: 4096 dimensional *relu7* layer outputs of both the VGG-M network [9] and the 16-layer VGG-D network [59].

FV/NetFV with CNNs. This denotes the method of [11] that builds a descriptor using FV pooling of CNN filter bank responses with 64 GMM components. One modification over [11] is that we first resize the image to 448×448 pixels, *i.e.*, twice the resolution the CNNs were trained on, and pool features from a *single-scale*. This leads to a slight

reduction in performance over the multi-scale approach. But we choose the single-scale setting because this keeps the feature extraction for all our methods identical making comparisons easier. We consider two representations based on the VGG-M and VGG-D networks. Unlike the earlier FC models, the features are extracted from the *relu5* and *relu5_3* layers of the VGG-M and VGG-D networks respectively.

VLAD/NetVLAD with CNNs. Similar to FV, this approach builds VLAD descriptors on CNN filter banks responses. We resize the image to 448×448 pixels before passing it through the network and aggregate the features obtained from the CNN using VLAD/NetVLAD pooling with 64 cluster centers. Identical to the FV models, we consider VLAD models with VGG-M and VGG-D networks truncated at *relu5* and *relu5_3* layers respectively.

BoVW/NetBoVW with CNNs. For BoVW we construct a vocabulary of 4096 words using k-means on top of the CNN features. For NetBoVW we use a 4096-way `softmax` layer as an approximation to the hard assignment. We use the same setting as FV and VLAD for feature extraction.

B-CNNs. These are models presented in Section 3 where features from two CNNs are pooled using an outer product. When the two CNNs are identical the model is a extension of O2P using deep features without the log-Euclidean normalization. We consider several B-CNNs – (i) one with two identical VGG-M networks truncated at the *relu5* layer, (ii) one with a VGG-D and VGG-M network truncated at the *relu5_3* and *relu5* layer respectively, and (iii) initialized with two identical VGG-D networks truncated at the *relu5_3* layer. Identical to the setting in FV and VLAD, the input images are resized to 448×448 and features are extracted using the two CNNs. The VGG-D network produces 28×28 output compared to 27×27 of the VGG-M network, so we downsample the VGG-D output by ignoring a row and column when combining it with the VGG-M output. The bilinear feature for all these models is of size 512×512. Note that the symmetric models, *i.e.*, option (i) and (iii), the two networks share all parameters (which is also the case when they are fine-tuned due to symmetry), so in practice these models have the same memory overhead and speed as a single network evaluation.

4.1.1 Fine-tuning

For fine-tuning we add k-way linear + softmax layer where k is the number of classes in the fine-grained dataset. The parameters of the linear layer are initialized randomly. We adopt a two step training procedure of [6] where we first train the linear 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 ($\eta = 0.001$). Across the datasets we found the hyperparameters for fine-tuning were fairly consistent. Although, the exact VLAD, FV, BoVW models cannot be directly fine-tuned, we report results using *indirect* fine-tuning where the networks are fine-tuned with FC layers. We found this improves accuracy. For example, the indirectly fine-tuned FV models with CNN features outperforms the multi-scale but not fine-tuned results reported in [11]. However, direct fine-tuning using NetFV is significantly better.

4.1.2 SVM training and evaluation

In all our experiments before and after fine-tuning, 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 ℓ_2 normalized, the optimal of C_{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 average the predictions of the image and its flipped copy. SVM training provides 1-3% improvement over logistic regression with the VGG-M networks, but provides *negligible* improvement with the VGG-D networks. Test time flipping improves performance by 0.5% on average for the VGG-M networks, while has *negligible* impact on the accuracy for the VGG-D networks. Performance is measured as the percentage of correctly classified images for all datasets.

4.2 Fine-grained recognition

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

CUB-200-2011 [67] 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 [42] 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.

Stanford cars dataset [33] 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 [64] is larger than the CUB dataset consisting of 48,562 images of 555 spices 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.

4.2.1 Bird species classification

Comparison to baselines. Table 2 "bird" column shows results on the CUB-200-2011 dataset. The end-to-end approximations of texture representations (NetBoWV, NetVLAD,

NetFV) improve significantly after fine-tuning. Exact models with indirect fine-tuning (*i.e.*, fine-tuned with FC layers) also improve, but the improvement is smaller (shown in gray italics in Table 2). With fine-tuning the single-scale FV models outperforms the multi-scale results reported in [11] – 49.9% using VGG-M and 66.7% using VGG-D network. B-CNNs offer the best accuracy across all models with the best performing model obtaining **84.1%** accuracy (VGG-M + VGG-D). The next best approach is the NetVLAD with 81.9% accuracy. We found that increasing the cluster centers does not improve performance of NetVLAD (see Section 5.1).

We also trained the B-CNN (VGG-M + VGG-D) model on the much larger *NABirds dataset*. For this model we skipped the SVM training step and report the accuracy using `softmax` layer predictions. This model achieves **79.4%** accuracy outperforming a fine-tuned VGG-D network that obtains 63.7% accuracy. Van Horn *et al*. [64] obtains **75%** accuracy using AlexNet and part annotations at test time, while the "neural activation constellations" approach [57] obtains **76.3%** accuracy using a GoogLeNet architecture [60].

Comparison to other techniques. Table 2 shows other top-performing methods on this dataset. The dataset also provides bounding-box and part annotations and techniques differ based on what annotations are used at training and test time (also shown in the Table). Two early methods that performed well when bounding-boxes are not available at test time are 73.9% of the "part-based R-CNN" [70] and 75.7% of the "pose-normalized CNN" [6]. These methods are based on AlexNet [34] and can be improved with deeper and more accurate networks such as the VGG-D. For example, the SPDA-CNN [69] trains better part detectors and feature representations using the VGG-D network and report 84.6% accuracy. Krause *et al*. [31] report 82.0% accuracy using a weakly-supervised method to learn part detectors, followed by the part-based analysis of [70] using the VGG-D network. However, our approach is simpler and faster since it does not rely on training and evaluating part detectors. The "cross-layer pooling" technique [38] that considers pairwise features extracted from two different layers of a CNN reports an accuracy of 73.5% using AlexNet and 77.0% accuracy with VGG-D. The approach uses bounding-boxes during training and testing.

One of the top-performing approaches that does not rely on additional annotations is the Spatial Transformer Networks [29]. It obtains **84.1%** accuracy using the batch-normalized Inception network [27]. The PD+SWFV-CNN approach combines unsupervised part detection with FV pooling of CNN features to obtain 83.6% accuracy, and **84.5%** accuracy with FC and FV pooling.

Since its publication, the B-CNN model has been improved by others in several ways. First, the *compact bilinear pooling* approach [19] was proposed to reduce the size of the bilinear features (we evaluate this in Section 5.1). Zhang *et al*. [72] combine B-CNNs with part annotations and improve results to **85.9%**. Moghimi *et al*. [44] boost B-CNNs trained on images of varying resolutions to obtain **86.2%** accuracy. Although not directly comparable, Krause *et al*. [32] show that by training on two orders of magnitude more labeled data obtained by querying category labels on search engines, the performance of deep architectures can be improved to 92.1%.

Fig. 4. Top six pairs of classes that are most confused with each other on the CUB dataset. In each row we show the images in the test set that were most confidently classified as the class in the other column.

Common mistakes. Figure 4 shows the top six pairs of classes that are confused by our fine-tuned B-CNN (VGG-M + VGG-D) model. The most confused pair of classes is "American crow" and "Common raven". The differences lie in the wing-spans, habitat, and voice, none of which are easy to measure from the image. Other confused classes are also similar – various Shrikes, Terns, Flycatchers, Cormorants and Gulls. We note that the dataset has an estimated 4.4% label noise hence some of these errors may come from incorrect labeling [64].

4.2.2 *Aircraft variant classification*

Comparison to baselines. The trends among the baselines are similar to those in birds with a few exceptions. The FV with SIFT is remarkably good (61.0%) and comparable to some of the CNN baselines. Compared to the birds, the effect of fine-tuning is significantly larger for models based on the VGG-D network suggesting a larger domain shift from the ImageNet dataset. As aircrafts appear mostly on the image center, cropping the central image improves the accuracy over our earlier work [37]. We resize the images into 512×512 and then crop the central 448×448 as input. This achieves the best performance of **86.9%** by the B-CNN (VDD-D) model. NetVLAD obtains 81.4% accuracy.

Comparison to other techniques. 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 2", *i.e.*, w/o bounding-boxes, at the FGComp 2013 challenge [2]. The best performing method [25] 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 $\rightarrow$ 1024), (ii) Spatial pyramid pooling ($1\times1 \rightarrow 1\times1 + 3\times1$), (iii) multiple SIFT variants, and (iv) multi-scale SIFT. Wang *et al*. [68] report **88.4%** accuracy by mining discriminative

TABLE 2
Per-image accuracy on the birds [67], aircrafts [42], and cars [33] datasets for various methods described in Section 4.1. We compare various texture representations and prior work (separated by a double line). The texture representations the first column lists the features used in the encoding followed by the pooling strategy. Features are extracted from the *relu5* and *relu5_3* layer of VGG-M and VGG-D networks respectively. FC pooling corresponds to fully-connected layers on top these intermediate layer features such that it corresponds to the penultimate layer of the original network. The second and third columns show additional annotations used during training and testing. Results are shown without and with domain-specific fine-tuning. Directly fine-tuning the approximate models leads to better performance than indirectly fine-tuning (shown in gray italics). Features are constructed from the corresponding layers of the fine-tuned FC-CNN models. B-CNN models achieve the best accuracy across texture representations. The first, second, and third best texture models are marked with red, blue and yellow colors respectively.

				birds		aircrafts		cars	
features	**train**	**test**	**encoding**	w/o ft	w/ ft	w/o ft	w/ ft	w/o ft	w/ ft
SIFT			FV	18.8	-	61.0	-	59.2	-
VGG-M (*relu5*)	*n/a*	*n/a*	FC	52.7	58.8	44.4	63.4	37.3	58.6
			BoVW	41.9	*43.8*	56.2	*60.1*	54.2	*58.4*
			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*
			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 (*relu5_3*)	*n/a*	*n/a*	FC	61.0	70.4	45.0	76.6	36.5	79.8
			BoVW	56.6	*58.8*	61.9	*71.3*	62.7	*73.9*
			NetBoVW	65.9	69.7	65.1	74.0	71.0	76.7
			VLAD	78.0	*79.0*	75.2	*80.6*	81.9	*85.6*
			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	77.7	86.9	82.9	90.6
VGG-M + VGG-D			B-CNN	80.1	84.1	78.0	86.6	83.9	91.3
VGG-D	*n/a*	*n/a*	PD+SWFV-CNN [73]	83.6		-		-	
VGG-D	*n/a*	*n/a*	PD+FC+SWFV-CNN [73]	84.5		-		-	
Inception-BN	*n/a*	*n/a*	STNs [29]	84.1		-		-	
VGG-D	Box	*n/a*	Krause *et al.* [31]	82.0		-		**92.6**	
VGG-D	Box+Part	Box	SPDA-CNN [69]	84.6		-		-	
VGG-D	Box	Box	CrossLayerPooling [38]	77.0		-		-	
VGG-D	Part	*n/a*	Zhang *et al.* [72]	85.9		-		-	
AlexNet	Box+Part	*n/a*	Part-based RCNN [70]	73.9		-		-	
AlexNet	Box+Part	*n/a*	Branson *et al.* [6]	75.7		-		-	
B-CNN (VGG-D)	*n/a*	*n/a*	BoostCNN [44]	**86.2**		**88.5**		92.1	
VGG-D	Box	Box	BoT [68]	-		88.4		92.5	
FV+SIFT	*n/a*	*n/a*	Gosselin *et al.* [25]	-		80.7		82.7	

patch triplets, but require bounding boxes during training and testing. Boosting B-CNNs [44] obtains the current state of the art with **88.5%** accuracy.

4.2.3 Car model classification

Comparison to baselines. FV with SIFT does well on this dataset achieving 59.2% accuracy. The effect of fine-tuning on cars is larger in comparison to birds and airplanes. Once again the B-CNNs outperform all the other baselines with the B-CNN (VGG-D + VGG-M) model achieving **91.3%** accuracy. NetVLAD obtains **88.6%** accuracy.

Comparison to other techniques. The best accuracy on this dataset is by Krause *et al.* [31] which obtains 92.6% accuracy. This is closely matched by **92.5%** accuracy of the discriminative patch triplets [68], and **92.1%** of Boosted B-CNNs [44]. Unlike the other approaches, Boosted B-CNNs do not rely on bounding-boxes at training time.

4.3 Texture and scene recognition

We experiment on three texture datasets – the *Describable Texture Dataset* (DTD) [10], *Flickr Material Dataset* (FMD) [56], and *KTH-TISP2-b* (KTH-T2b) [7]. 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

TABLE 3
Mean per-class accuracy on DTD, FMD, KTH-T2b and MIT indoor datasets using FV and B-CNN representations constructed on top of *relu5_3* layer outputs of the 16-layer VGG-D network [59]. Results for input images at different scales $s = 1$, $s = 2$ and ms correspond to a size of 224×224, 448×448 and multiple sizes respectively.

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

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 [51]. 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 to the prior state-of-the-art approach of FV pooling of CNN features [11] using the VGG-D network. These results are without fine-tuning. On the MIT indoor dataset fine-tuning B-CNNs leads to a small improvement $72.8\% \rightarrow 73.8\%$ using *relu5_3* at $s = 1$, while on the other datasets the improvements were negligible, likely due to the relatively small size of these datasets. Table 3 shows the results obtained by features from a single scale and features from multiple scales $2^s, s \in \{1.5\text{:-}0.5\text{:-}3\}$ relative to the 224×224 image using B-CNN and FV representations. We discard scales for which the image is smaller than the size of the receptive fields of the filters, or larger than 1024^2 pixels for efficiency. Across all scales of the input image the performance of the two approaches are identical. Multiple scales consistently lead to an improvement in accuracy. The multi-scale FV results reported here are comparable ($\pm 1\%$) to the results reported in Cimpoi *et al*. [11] for all datasets except KTH-T2b (-4%). These differences are 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 B-CNNs are comparable to the FV pooling for texture recognition. One drawback is that the FV features with 64 GMM components has smaller in size (64×2×512) than the bilinear features (512×512). However, bilinear features are highly redundant and their dimensionality can be reduced by an order of magnitude without loss in performance (see Section 5.1).

5 Analysis of Bilinear CNNs

5.1 Dimensionality reduction

The outer product of CNN features generates very high dimensional image descriptors, *e.g.*, 262K for the B-CNN models in Table 2. Our earlier work [36] 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 [30] has also shown that in the context of SIFT-based FV and VLAD, highly compact representations can be obtained.

In this section we investigate the trade-off between accuracy and feature dimension for various texture models proposed in Section 3 for fine-grained recognition. For NetVLAD and NetFV the feature dimension can be varied by changing the number of cluster centers. For B-CNNs, consider the case where the outer product is computed among features $\mathbf{x}$ and $\mathbf{y}$. There are several strategies for reducing the feature dimension:

(1) Projecting the outer product into a lower dimensional space, *i.e.*, $\Phi(\mathbf{x}, \mathbf{y}) = \texttt{vec}(\mathbf{x}^T\mathbf{y})\mathbf{P}$, where $\mathbf{P}$ is a projection matrix and the $\texttt{vec}$ operator reshapes the matrix into a vector.
(2) Projecting both the features into a lower-dimensional space and computing outer product, $\Phi(\mathbf{x}, \mathbf{y}) = (\mathbf{x}\mathbf{A})^T(\mathbf{y}\mathbf{B})$, where $\mathbf{A}$, $\mathbf{B}$ are projection matrices.
(3) Projecting one of the features into a lower-dimensional space and computing the outer product, *i.e.*, by setting $\mathbf{B}$ to an identity matrix in the previous approach.

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 [37], and more recently in [19]. We believe this is because after the PCA rotation, the features are no longer correlated across dimensions. Remarkably, reducing the dimension of only one feature using PCA (third option) works well in practice. 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 [37] to reduce the feature size. It breaks the symmetry of the features when identical networks are used and is an example of a partially shared feature pipeline (Figure 3b). It also resembles the computations of VLAD and FV representations where both f_A and f_B are based on the same underlying feature.

The accuracy as a function of feature dimension shown in Figure 5 for NetFV and NetVLAD. These results are obtained by varying the number of cluster centers. The results indicate the performance of NetVLAD and NetFV do not improve with more cluster centers beyond 32.

Figure 6 shows the same for B-CNN features. We also compare our PCA approach to the recently-proposed Compact Bilinear Pooling (CBP) technique [19]. CBP approximates the outer product using a product of sparse linear projections of features with a Tensor Sketch [48]. The performance of the full model with 512×512 dimensions with and without fine-tuning is shown as a straight line. On birds and aircrafts the dimensionality can be reduced by 16× (*i.e.*, to 32×512) with a less than 1% loss in accuracy. In comparison, NetVLAD with the same feature size (*i.e.*, with 32 components) is about 3-4% less accurate. Overall, for a given

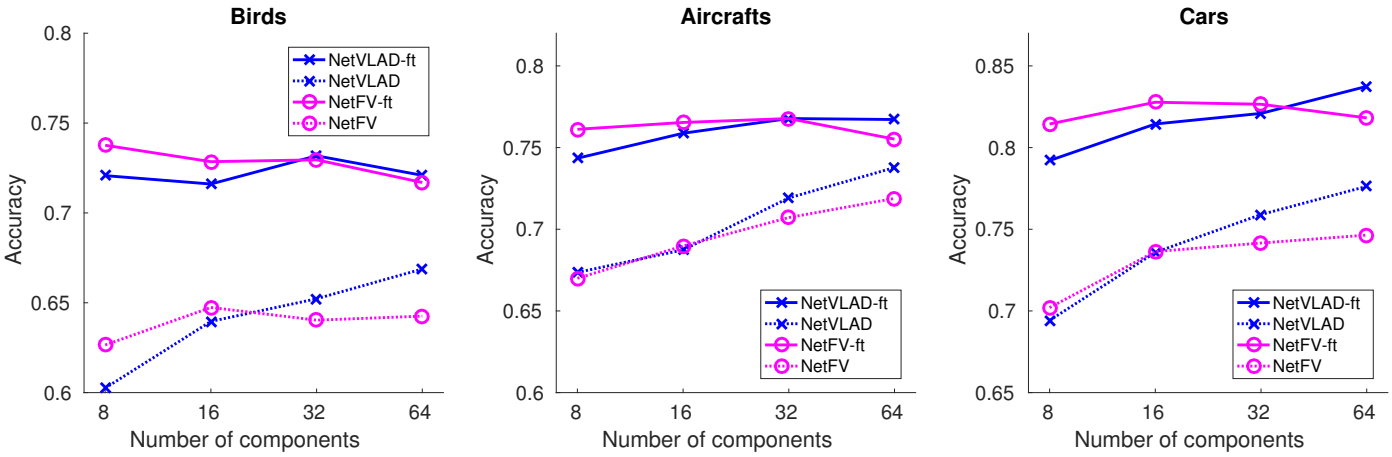


Fig. 5. Performance of NetVLAD and NetFV models encoding VGG-M *relu5* features with different number of cluster centers on fine-grained datasets before (dashed lines) and after (solid lines) fine-tuning. Given the same number of cluster centers, the feature dimension of NetFV representation is twice as large as NetVLAD.

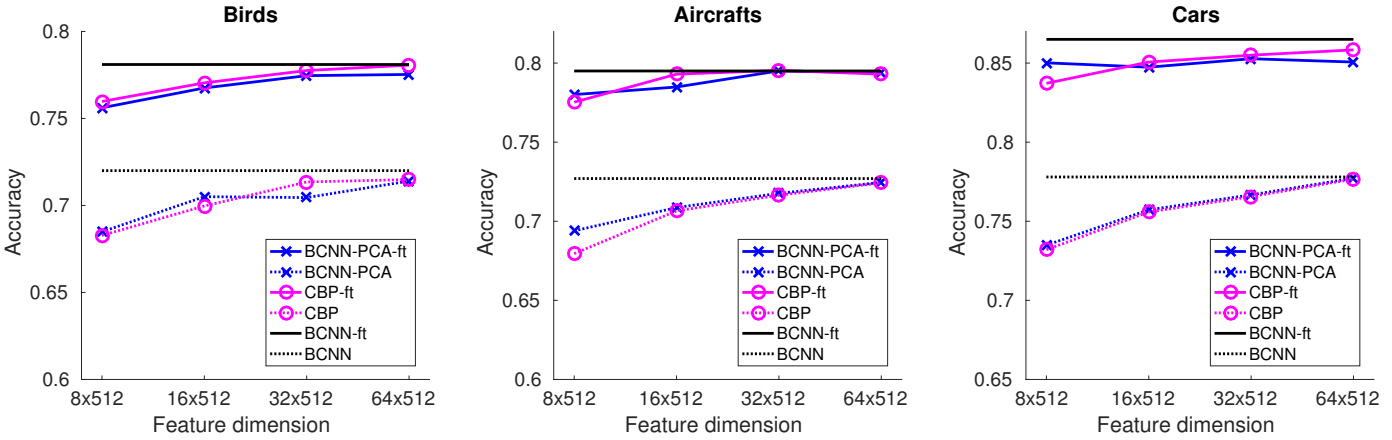


Fig. 6. Performance of B-CNNs using VGG-M *relu5* features as function of feature dimension before (dashed lines) and after (solid lines) fine-tuning. One of the 512 dimensional feature is projected using PCA to k dimensions leading to a outer product of size $k \times 512$ (see Section 5.1 for details). The performance using Compact Bilinear Pooling (CBP) [19] and the full 512×512-dimensional model is shown in red and black respectively.

budget of dimensions the projected B-CNNs outperform NetVLAD and NetFV representations. The PCA approach is slightly worse than CBP. However, one advantage is that PCA can be implemented as a dense matrix multiplication which is empirically $1.5\times$ faster than CBP which involves computing Fourier transforms and their inverses.

5.2 Training B-CNNs on ImageNet LSVRC

We evaluate B-CNNs trained *from scratch* on the ImageNet LSRVC 2012 dataset [54]. In particular, we train a B-CNN with a *relu5* layer output of VGG-M network and compare it to the standard VGG-M network. This allows a direct comparison of bilinear pooling with FC pooling since the rest of the architecture is identical in both networks. Additionally, we compare the effect of implicit translational invariance obtained in CNNs by spatially "jittering" the data, as well a explicit translation invariance of B-CNNs due to the orderless pooling.

We train the two networks to classify 224×224 images with different amounts of spatial jittering – "f1" for flip, "f5" for flip + 5 translations, and "f25" for flip + 25 translations. Training is done with stochastic sampling where one of the jittered copies is randomly selected for each example. The 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, and continue till no further improvement is observed.

Table 4 shows the "top1" and "top5" validation errors for B-CNN and VGG-M. 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 [9]. The VGG-M model achieves 46.4% top1 error with flip augmentation during training. The performance improves significantly to 39.6% with f25 augmentation. B-CNN achieves 38.7% top1 error with f1 augmentation, outperforming VGG-M trained with f25 augmentation. The results show that B-CNN is discriminative, robust to translation and that explicit translation invariance is more effective. This trend is also reflected in the latest deep architectures such as Residual Networks [26] that replace the fully-connected layers with global pooling layers.

TABLE 4
Accuracy on the ILSVRC 2014 *validation* set using bilinear and FC pooling on top of *relu5* layer output of a VGG-M network. Both networks are trained from scratch on the ILSVRC 2012 training set with varying amounts of data augmentation.

	Bilinear pooling			FC pooling	
data aug.	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

5.3 Visualizing learned models

Top activations of B-CNN units. Figure 7 shows the top activations of several units of the *relu5_3* layer of VGG-D and the *relu5* layer of VGG-M network for the fine-tuned B-CNN (VGG-D + VGG-M) model. Both these networks contain units that activate strongly on highly localized features. For example, the last row of VGG-D detects "tufted heads", while the fourth row in the same column detects a "red-yellow stripe" on birds. Similarly for airplanes, the units localize different types of windows, noses, vertical stabilizers, with some specializing in detecting particular airliner logos. For cars, units activate on different kinds of head/tail lights, wheels, *etc.*

Inverting categories. To understand the properties learned by the B-CNNs we visualize pre-images of a category by "inversion". We use the framework of Mahendran and Vedaldi [41] to generate an image that produces a high score for a target category $\hat{C}$ based on a B-CNN classifier. Specifically, for an image $\mathbf{x}$ and a layer $r_i, i = 1, \ldots, n$, we compute the bilinear features B_{r_i} using a B-CNN. Let C_{r_i} be the class prediction probabilities obtained using linear classifier trained on B_{r_i} in a supervised manner. We obtain an image that maximizes a target label $\hat{C}$ by solving the following optimization:

$$\min_{\mathbf{x}} \sum_{i=1}^{m} L\left(C_{r_i}, \hat{C}\right) + \gamma \Gamma(\mathbf{x}). \tag{5}$$

Here, L is a loss function such as the *negative log-likelihood* of the label $\hat{C}$ and γ is a tradeoff parameter. The image prior $\Gamma(\mathbf{x})$ encourages the smoothness of output image. We use the TV_β norm with $\beta = 2$:

$$\Gamma(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2\right)^{\frac{\beta}{2}}. \tag{6}$$

The exponent $\beta = 2$ was empirically found to lead to fewer "spike" artifacts in the optimization [41]. We use B-CNNs based on the VGG-D network. In our experiments, an input image is resized to 224×224 pixels before computing the target bilinear features. We solve for $\mathbf{x} \in \mathbb{R}^{224\times224\times3}$ on the optimization. The lower resolution 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 *relu2_2*, *relu3_3*, *relu4_3*, *relu5_3*. We use L-BFGS for optimization and compute the gradients of the objective with respect to $\mathbf{x}$ using back-propagation. The hyperparameter $\gamma = 10^{-8}$ was found empirically to lead to good inverse images. We also refer readers to our earlier work [36] and

Fig. 7. Patches with the highest activation for several filters of the fine-tuned B-CNN (VGG-D + VGG-M) model on birds, aircrafts and cars dataset. These visualizations indicate that network units activate on highly localized attributes of objects that capture color, texture, and shape patterns.

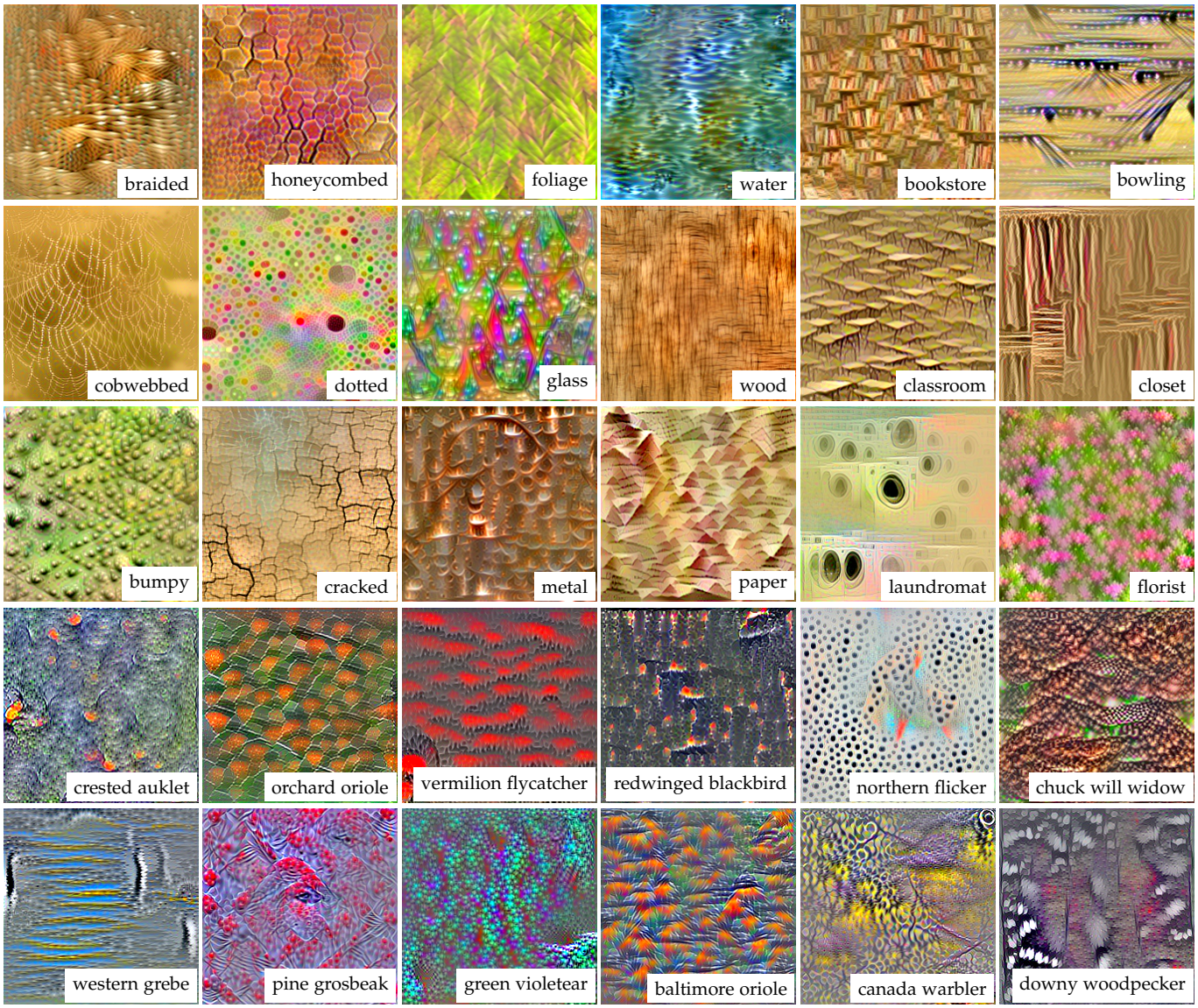


Fig. 8. Visualizing various categories by inverting the B-CNN based on VGG-D network trained on DTD [10], FMD [56], MIT Indoor dataset [51] (first three rows, two columns each from left to right), and the CUB dataset [67] (last two rows, all columns). *Best viewed in color and with zoom.*

more recent work [61], where this framework was applied for texture synthesis and style transfer using attributes.

Figure 8 shows some inverse images for various categories for the DTD, FMD, MIT indoor, and CUB-200-2011 dataset. These images reveal how B-CNNs represents various categories as textures. 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. The inverse images of the MIT indoor dataset reveal key properties of a category – a *bookstore* has a racks of books, a *laundromat* has laundry machines at various scales and locations. The inverse images of various bird species capture distinctive colors and patterns on their bodies. Figure 9 visualizes reconstructions by incrementally adding layers in the bilinear representation. Even though the *relu5_3* layer provides the best recognition accuracy, simply using that layer did not produce good inverse images as the color information was missing.

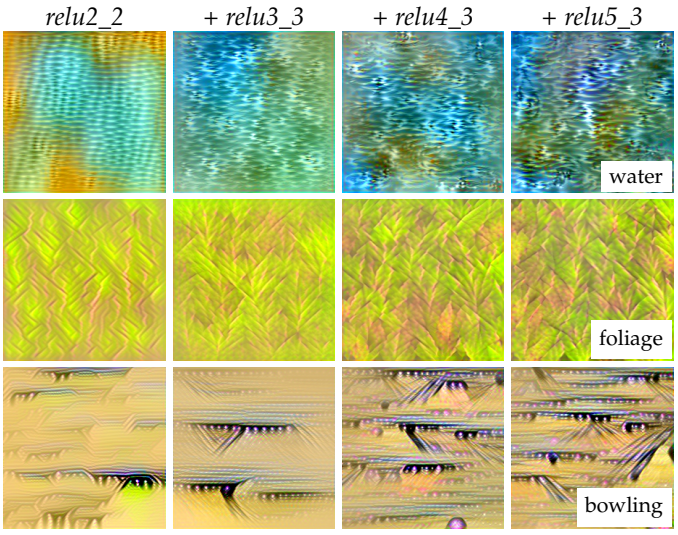


Fig. 9. Inverse images obtained from a multilayer B-CNN. From left to right different layers (shown on top) are added one by one.

6 Conclusion

We presented the B-CNN architecture that aggregates second-order statistics of CNN activations resulting in an orderless representation of an image. These networks can be trained in an end-to-end manner allowing both training from scratch on large datasets, and domain-specific fine-tuning for transfer learning. Moreover, these models are fairly efficient, processing 448×448 resolution images at 30-100 FPS on a NVIDIA Titan X GPU. We also compared B-CNNs to both exact and approximate variants of deep texture representations and studied the accuracy and memory trade-offs they offer. The main conclusion was that variants of outer-product representations are highly effective at various fine-trained, texture, and scene recognition tasks. Moreover, these representations are redundant and in most cases their dimension can be reduced by an order of magnitude without significant loss in accuracy. A visualization of B-CNNs showed that these models effectively represent objects as *texture* and their units are correlated with localized attributes useful for fine-grained recognition.

Acknowledgement: This research was supported in part by the National Science Foundation grant IIS-1617917, a faculty gift from Facebook, and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) under contract number 2014-14071600010. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon. The experiments were performed using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Tech Collaborative and GPUs donated by NVIDIA.

References

[1] https://www.allaboutbirds.org/guide/Ring-billed_Gull/id. Accessed: 2017-03-28. 1

[2] https://sites.google.com/site/fgcomp2013/results. Accessed: 2017-03-30. 7

[3] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5

[4] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In *International Conference on Learning Representations*, 2015. 2

[5] L. Bourdev, S. Maji, and J. Malik. Describing people: A poselet-based approach to attribute classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2011. 2

[6] 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. 2, 6, 7, 8

[7] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *IEEE International Conference on Computer Vision (ICCV)*, 2005. 8

[8] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision (ECCV)*. 2012. 1, 2, 4

[9] 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. 2, 5, 10

[10] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 3, 8, 12

[11] 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, 2, 4, 5, 6, 7, 9

[12] 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*, 2004. 1, 4

[13] 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, 3

[14] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The PASCAL visual obiect classes challenge 2007 (VOC2007) results. Technical report, Pascal Challenge, 2007. 2

[15] 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 *IEEE International Conference on Computer Vision (ICCV)*, 2011. 2

[16] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[17] K. Fragkiadaki, P. Arbeláez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3

[18] 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. 3

[19] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3, 7, 9, 10

[20] 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. 2

[21] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[22] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, 2012. 3

[23] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2, 3

[24] 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, 2

[25] P.-H. Gosselin, N. Murray, H. Jégou, and F. Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49:92–98, 2014. 2, 4, 7, 8

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 10

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

[28] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 5

[29] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*. 2015. 2, 7, 8

[30] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 4, 9

[31] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 7, 8

[32] 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. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 7

[33] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *3D Representation and Recognition Workshop, at ICCV*, 2013. 1, 6, 8

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 2, 7

[35] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001. 2

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

[37] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN Models for Fine-grained Visual Recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 3, 4, 5, 7, 9

[38] L. Liu, C. Shen, and A. van den Hengel. Cross-convolutional-layer pooling for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 2, 7, 8

[39] 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. 3

[40] D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, 1999. 1

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

[42] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 1, 6, 8

[43] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*. 2014. 2

[44] M. Moghimi, S. Belongie, M. Saberian, J. Yang, N. Vasconcelos, and L.-J. Li. Boosted convolutional neural networks. In *British Machine Vision Conference (BMVC)*, 2016. 2, 7, 8

[45] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 4

[46] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 2

[47] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, 2010. 4, 5

[48] 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. 3, 9

[49] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In *Advances in Neural Information Processing Systems*. 2009. 3

[50] J. Portilla and E. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000. 2

[51] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 9, 12

[52] A. S. Razavin, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *DeepVision workshop*, 2014. 2, 3

[53] 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*. 2015. 3

[54] 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. 1, 10

[55] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001. 3, 4

[56] L. Sharan, R. Rosenholtz, and E. H. Adelson. Material perceprion: What can you see in a brief glance? *Journal of Vision*, 9:784(8), 2009. 8, 12

[57] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 7

[58] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014. 3

[59] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 2, 5, 9

[60] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 7

[61] T. G. T. Irmer and S. Maji. Texture attribute synthesis and transfer using feed-forward CNNs. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 12

[62] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000. 3

[63] I. Ustyuzhaninov, W. Brendel, L. A. Gatys, and M. Bethge. Texture synthesis using shallow convolutional networks with random filters. *arXiv preprint arXiv:1606.00021*, 2016. 2

[64] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 6, 7

[65] A. Vedaldi and B. Fulkerson. VLFeat: an open and portable library of computer vision algorithms. In *ACM International Conference on Multimedia*, 2010. 5

[66] A. Vedaldi and K. Lenc. MatConvNet – Convolutional Neural Networks for MATLAB. In *ACM International Conference on Multimedia*, 2015. 1, 3

[67] C. 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. 1, 2, 6, 8, 12

[68] Y. Wang, J. Choi, V. Morariu, and L. S. Davis. Mining discriminative triplets of patches for fine-grained classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7, 8

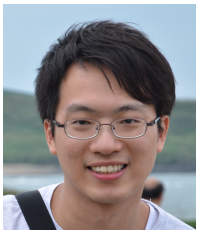
[69] H. Zhang, T. Xu, M. Elhoseiny, X. Huang, S. Zhang, A. Elgammal, and D. Metaxas. SPDA-CNN: unifying semantic part detection and abstraction for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 7, 8

[70] 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. 2, 7, 8

[71] N. Zhang, R. Farrell, and T. Darrell. Pose pooling kernels for sub-category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[72] N. Zhang, E. Shelhamer, Y. Gao, and T. Darrell. Fine-grained pose prediction, normalization, and recognition. In *Workshop at International Conference on Learning Representations*, 2016. 7, 8

[73] X. Zhang, H. Xiong, W. Zhou, W. Lin, and Q. Tian. Picking deep filter responses for fine-grained image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8



Tsung-Yu Lin received his BS and MS degrees in computer science from National Tsing Hua University in 2008 and 2010 respectively. He worked at Academia Sinica, a governmental research institute in Taiwan as research assistant after he obtained MS degree. He is currently a PhD student in the College of Information and Computer Sciences at University of Massachusetts Amherst. He is interesting in applying machine learning techniques to computer vision problems, especially image classification and segmentation.

Aruni RoyChowdhury is a PhD student in the College of Information and Computer Sciences, University of Massachusetts, Amherst. His research interests are in Computer Vision, focusing on applications of deep learning to face recognition and fine-grained recognition. Previously he has worked on scene text detection and handwriting recognition at the Indian Statistical Institute, Kolkata. He obtained his MS degree (2016) from the University of Massachusetts, Amherst and his BTech (2013) from Heritage Institute of Technology, Kolkata in India.

Subhransu Maji is an Assistant Professor in the College of Information and Computer Sciences at the University of Massachusetts, Amherst. Previously, he was a Research Assistant Professor at TTI Chicago, a philanthropically endowed academic computer science institute in the University of Chicago campus. He obtained his Ph.D. from the University of California, Berkeley in 2011, and B.Tech. in Computer Science and Engineering from IIT Kanpur in 2006. His research focuses on developing visual recognition architectures with an eye for efficiency and accuracy.