CrossMark

Deep Filter Banks for Texture Recognition, Description, and Segmentation

Mircea Cimpoi[1] · **Subhransu Maji[2]** · **Iasonas Kokkinos[3]** · **Andrea Vedaldi[1]**

Abstract Visual textures have played a key role in image understanding because they convey important semantics of images, and because texture representations that pool local image descriptors in an orderless manner have had a tremendous impact in diverse applications. In this paper we make several contributions to texture understanding. First, instead of focusing on texture instance and material category recognition, we propose a human-interpretable vocabulary of texture attributes to describe common texture patterns, complemented by a new *describable texture dataset* for benchmarking. Second, we look at the problem of recognizing materials and texture attributes in realistic imaging conditions, including when textures appear in clutter, developing corresponding benchmarks on top of the recently proposed OpenSurfaces dataset. Third, we revisit classic texture represenations, including bag-of-visual-words and the Fisher vectors, in the context of deep learning and show that these have excellent efficiency and generalization properties if the convolutional layers of a deep model are used as filter banks. We obtain in this manner state-of-the-art performance in numerous datasets well beyond textures, an efficient method to apply deep features to image regions, as well as benefit in transferring features from one domain to another.

1 Introduction

Visual representations based on orderless aggregations of local features, which were originally developed as texture descriptors, have had a widespread influence in image understanding. These models include cornerstones such as the histograms of vector quantized filter responses of Leung and Malik (1996) and later generalizations such as the bag-of-visual-words model of Csurka et al. (2004) and the Fisher vector of Perronnin and Dance (2007). These and other texture models have been successfully applied to a huge variety of visual domains, including problems closer to "texture understanding" such as material recognition, as well as domains such as object categorization and face identification that share little of the appearance of textures.

This paper makes three contributions to texture understanding. The first one is to add a new *semantic dimension* to the problem. We depart from most of the previous works on visual textures, which focused on texture identification and material recognition, and look instead at the problem of *describing generic texture patterns*. We do so by developing a vocabulary of forty-seven *texture attributes* that describe a wide range of texture patterns; we also introduce a large dataset annotated with these attributes which we call the *describable texture dataset* (Sect. 2). We then study whether

✉ Mircea Cimpoi
mircea@robots.ox.ac.uk

Subhransu Maji
smaji@cs.umass.edu

Iasonas Kokkinos
iasonas.kokkinos@ecp.fr

Andrea Vedaldi
vedaldi@robots.ox.ac.uk

[1] University of Oxford, Oxford, UK

[2] University of Massachusetts Amherst, Amherst, USA

[3] CentraleSupélec/INRIA-Saclay, Palaiseau, France

🐾 Springer

texture attributes can be reliably estimated from images, and for what tasks are they useful. We demonstrate in particular two applications (Sect. 8.1): the first one is to use texture attributes as dimensions to organise large collections of texture patterns, such as textile, wallpapers, and construction materials for search and retrieval. The second one is to use texture attributes as a compact basis of visual descriptors applicable to other tasks such as material recognition.

The second contribution of the paper is to introduce new *data and benchmarks* to study texture recognition in realistic settings. While most of the earlier work on texture recognition was carried out in carefully controlled conditions, more recent benchmarks such as the Flickr material dataset (FMD) (Sharan et al. 2009) have emphasized the importance of testing algorithms "in the wild", for example on Internet images. However, even these datasets are somewhat removed from practical applications as they assume that textures fill the field of view, whereas in applications they are often observed in clutter. Here we leverage the excellent OpenSurfaces dataset (Bell et al. (2013)) to create novel benchmarks for materials and texture attributes where textures appear both in the wild and in clutter (Sect. 3), and demonstrate promising recognition results in these challenging conditions. In Bell et al. (2015) the same authors have also investigated material recognition using OpenSurfaces.

The third contribution is technical and revisits classical ideas in texture modeling in the light of modern local feature descriptors and pooling encoders. While texture representations were extensively used in most areas of image understanding, since the breakthrough work of Krizhevsky et al. (2012) they have been replaced by deep Convolutional Neural Networks (CNNs). Often CNNs are applied to a problem by using transfer learning, in the sense that the network is first trained on a large-scale image classification task such as the ImageNet ILSVRC challenge (Deng et al. (2009)), and then applied to another domain by exposing the output of a so-called "fully connected layer" as a general-purpose image representation. In this work we illustrate the many benefits of truncating these CNNs earlier, at the level of the convolutional layers (Sect. 4). In this manner, one obtains powerful *local image descriptors* that, combined with traditional pooling encoders developed for texture representations, result in state-of-the-art recognition accuracy in a diverse set of visual domains, from material and texture attribute recognition, to coarse and fine grained object categorization and scene classification. We show that a benefit of this approach is that features transfer easily across domains even without fine-tuning the CNN on the target problem. Furthermore, pooling allows us to efficiently evaluate descriptors in image subregions, a fact that we exploit to recognize local image regions without recomputing CNN features from scratch.

A symmetric approach, using SIFT as local features and the IFV followed by fully-connected layers from a deep neural network as a pooling mechanism, was proposed in Perronnin and Larlus (2015), obtaining similar results on VOC07.

This paper is the archival version of two previous publications Cimpoi et al. (2014) and Cimpoi et al. (2015). Compared to these two papers, this new version adds a significant number of new experiments and a substantial amount of new discussion.

The code and data for this paper are available on the project page, at http://www.robots.ox.ac.uk/~vgg/research/deeptex.

2 Describing Textures with Attributes

This section looks at the problem of automatically describing texture patterns using a general-purpose vocabulary of human-interpretable texture attributes, in a manner similar to how we can vividly characterize the textures shown in Fig. 1. The goal is to design algorithms capable of generating and understanding texture descriptions involving a *combination of describable attributes* for each texture. Visual attributes have been extensively used in *search*, to understand complex user queries, in *learning*, to port textual information back to the visual domain, and in image *description*, to produce richer accounts of the content of images. Textural properties are an important component of the semantics of images, particularly for objects that are best characterized by a pattern, such as a scarf or the wings of a butterfly (Wang et al. 2009). Nevertheless, the attributes of visual textures have been investigated only tangentially so far. Our aim is to fill this gap.

Our first contribution is to introduce the **describable textures dataset** (DTD) (Cimpoi et al. 2014), a collection

porous, dotted, freckled, honeycombed

braided, interlaced, knitted, woven, zigzagged

scaly, crosshatched, flecked, studded, waffled

wrinkled, crystalline, flecked, smeared

cracked, pitted, studded

fibrous, freckled, interlaced, smeared, swirly

Fig. 1 We address the problem of *describing textures* by associating to them a collection of attributes. Our goal is to understand and generate automatically human-interpretable descriptions such as the examples above
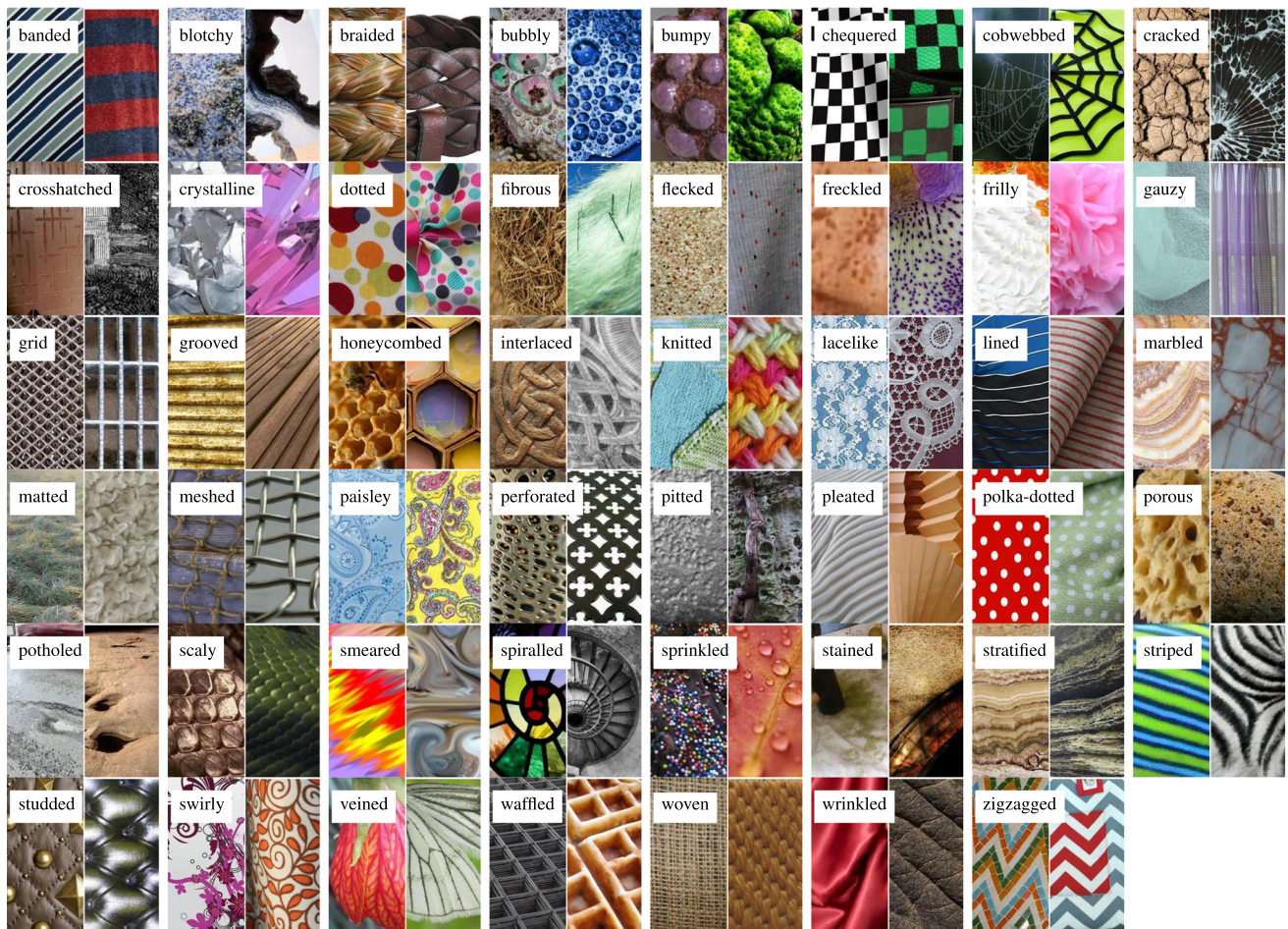
Fig. 2 The 47 texture words in the *describable texture dataset* introduced in this paper. Two examples of each attribute are shown to illustrate the significant amount of variability in the data

of real-world texture images annotated with one or more adjectives selected in a vocabulary of forty-seven English words. These adjectives, or *describable texture attributes*, are illustrated in Fig. 2 and include words such as *banded*, *cobwebbed*, *freckled*, *knitted*, and *zigzagged*. Sect. 2.1 describes this data in more detail. Sect. 2.2 discusses the technical challenges we addressed while designing and collecting DTD, including how the forty-seven texture attributes were selected and how the problem of collecting numerous attributes for a vast number of images was addressed. Sect. 2.3 defines a number of benchmark tasks in DTD. Finally, Sect. 2.5 relates DTD to existing texture datasets.

2.1 The Describable Texture Dataset

DTD investigates the problem of *texture description*, understood as the recognition of describable texture attributes. This problem is complementary to standard texture analysis tasks such as texture identification and material recognition for the following reasons. While describable attributes are correlated with materials, attributes do not imply materials (e.g. *veined* may equally apply to leaves or marble) and materials do not imply attributes (not all marbles are *veined*). This distinction is further elaborated in Sect. 2.4.

Describable attributes can be *combined* to create rich descriptions (Fig. 3; marble can be *veined*, *stratified* and *cracked* at the same time), whereas a typical assumption is that textures are made of a single material. Describable attributes are *subjective* properties that depend on the imaged object as well as on human judgements, whereas materials are objective. In short, attributes capture properties of textures complementary to materials, supporting human-centric tasks where describing textures is important. At the same time, we will show that texture attributes are also helpful in material recognition (Sect. 8.1).

DTD contains **textures in the wild**, *i.e.* texture images extracted from the web rather than captured or generated in a controlled setting. Textures fill the entire image in order to allow studying the problem of texture description independently of texture segmentation, which is instead addressed in
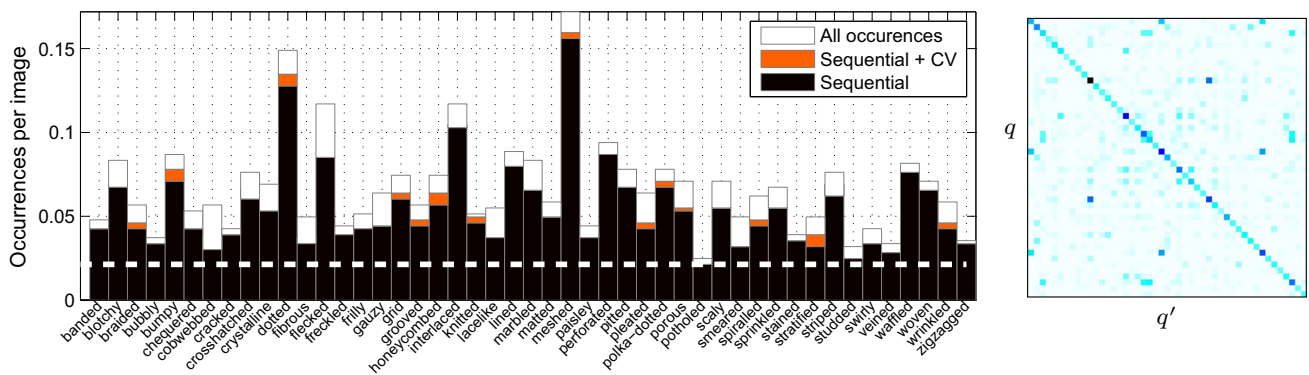
Fig. 3 Quality of sequential joint annotations. Each *bar* shows the average number of occurrences of a given attribute in a DTD image. The *horizontal dashed line* corresponds to a frequency of 1/47, the minimum given the design of DTD (Sect. 2.2). The *black portion* of each bar is the amount of attributes discovered by the sequential procedure, using only ten annotations per image (about one fifth of the effort required for exhaustive annotation). The *orange portion* shows the additional recall obtained by integrating cross-validation in the process. *Right:* co-occurrence of attributes. The *matrix* shows the joint probability $p(q, q')$ of two attributes occurring together (rows and columns are sorted in the same way as the *left* image) (Color figure online)

Sect. 3. With 5640 annotated texture images, this dataset aims at supporting real-world applications were the recognition of texture properties is a key component. Collecting images from the Internet is a common approach in categorization and object recognition, and was adopted in material recognition in FMD. This choice trades-off the systematic sampling of illumination and viewpoint variations existing in datasets such as CUReT, KTH-TIPS, Outex, and Drexel to capture real-world variations, reducing the gap with applications. Furthermore, DTD captures empirically human judgements regarding the invariance of describable texture attributes; this invariance is not necessarily reflected in material properties.

2.2 Dataset Design and Collection

This section discusses how DTD was designed and collected, including: selecting the 47 attributes, finding at least 120 representative images for each attribute, and collecting all the attribute labels for each image in the dataset.

2.2.1 Selecting the Describable Attributes

Psychological experiments suggest that, while there are a few hundred words that people commonly use to describe textures, this vocabulary is redundant and can be reduced to a much smaller number of representative words. Our starting point is the list of 98 words identified by Bhushan et al. (1997). Their seminal work aimed to achieve for texture recognition the same that color words have achieved for describing color spaces (Berlin and Kay 1991). However, their work mainly focuses on the cognitive aspects of texture perception, including perceptual similarity and the identification of directions of perceptual texture variabil-

ity. Since our interest is in the visual aspects of texture, words such as "corrugated" that are more related to surface shape or haptic properties were ignored. Other words such as "messy" that are highly subjective and do not necessarily correspond to well defined visual features were also ignored. After this screening phase we analyzed the remaining words and merged similar ones such as "coiled", "spiraled" and "corkscrewed" into a single term. This resulted in a set of 47 words, illustrated in Fig. 2.

2.2.2 Bootstrapping the Key Images

Given the 47 attributes, the next step consisted in collecting a sufficient number (120) of example images representative of each attribute. Initially, a large initial pool of about a hundred-thousand images in total was downloaded from Google and Flickr by entering the attributes and related terms as search queries. Then Amazon Mechanical Turk (AMT) was used to remove low resolution, poor quality, watermarked images, or images that were not almost entirely filled with a texture. Next, detailed annotation instructions were created for each of the 47 attributes, including a dictionary definition of each concept and examples of textures that did and did not match the concept. Votes from three AMT annotators were collected for the candidate images of each attribute and a shortlist of about 200 highly-voted images was further manually checked by the authors to eliminate remaining errors. The result was a selection of 120 *key representative images* for each attribute.

2.2.3 Sequential Joint Annotation

So far only the key attribute of each image is known while any of the remaining 46 attributes may apply as well.

Exhaustively collecting annotations for 46 attributes and 5640 texture images is fairly expensive. To reduce this cost we propose to exploit the correlation and sparsity of the attribute occurrences (Fig. 3). For each attribute q, twelve key images are annotated exhaustively and used to estimate the probability $p(q'|q)$ that *another* attribute q' could co-exist with q. Then for the remaining key images of attribute q, only annotations for attributes q' with non negligible probability are collected, assuming that the remaining attributes would not apply. In practice, this requires annotating around 10 attributes per texture instance, instead of 47. This procedure occasionally misses attribute annotations; Fig. 3 evaluates attribute recall by 12-fold cross-validation on the 12 exhaustive annotations for a fixed budget of collecting 10 annotations per image.

A further refinement is to suggest which attributes q' to annotate not just based on the prior $p(q'|q)$, but also based on the appearance of an image $\mathbf{x}_i$. This was done by using the attribute classifier learned in Sect. 6; after Platt's calibration (Platt 2000) on a held-out test set, the classifier score $c_{q'}(\mathbf{x}_i) \in \mathbb{R}$ is transformed in a probability $p(q'|\mathbf{x}_i) = \sigma(c_{q'}(\mathbf{x}))$ where $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid function. By construction, Platt's calibration reflects the prior probability $p(q') \approx p_0 = 1/47$ of q' on the validation set. To reflect the probability $p(q'|q)$ instead, the score is adjusted as

$$p(q'|\ell_i, q) \propto \sigma(c_{q'}(\ell_i)) \times \frac{p(q'|q)}{1 - p(q'|q)} \times \frac{1 - p_0}{p_0}$$

and used to find which attributes should be annotated for each image. As shown in Fig. 3, for a fixed annotation budget this method increases attribute recall.

Overall, with roughly 10 annotations per image it was possible to recover all of the attributes for at least 75 % of the images, and miss one out of four (on average) for another 20 %, while keeping the annotation cost to a reasonable level. To put this in perspective, directly annotating the 5640 images for 46 attributes and collecting five annotations per attributed would have required 1.2M binary annotations, i.e. roughly 12K USD at the very low rate of 1¢ per annotation. Using the proposed method, the cost would have been 546 USD. In practice, we spent around 2.5K USD in order to pay annotators better as well as to account for occasional errors in setting up experiments and the fact that, as explained above, bootstrapping still relies on exhaustive annotations for a subset of the data.

2.3 Benchmark Tasks

DTD is designed as a public benchmark. The data, including images, annotations, and splits, is available on the web at http://www.robots.ox.ac.uk/~vgg/data/dtd, along with code for evaluation and reproducing the results in Sect. 6.

DTD defines two challenges. The first one, denoted DTD, is the *prediction of key attributes*, where each image is assigned a single label corresponding to the key attribute defined above. The second one, denoted DTD-J, is the joint *prediction of multiple attributes*. In this case each image is assigned one or more labels, corresponding to all the attributes that apply to that image.

The first task is evaluated both in term of classification accuracy (acc) and in term of mean average precision (mAP), while the second task only in term of mAP due to the possibility of multiple labels. The classification accuracy is normalized per class: if $\hat{c}(\mathbf{x}), c(\mathbf{x}) \in \{1, \ldots, C\}$ are respectively the predicted and ground-truth label of image $\mathbf{x}$, accuracy is defined as

$$\text{acc}(\hat{c}) = \frac{1}{C} \sum_{\bar{c}=1}^{C} \frac{|\{\mathbf{x} : c(\mathbf{x}) = \bar{c} \wedge \hat{c}(\mathbf{x}) = \bar{c}\}|}{|\{\mathbf{x} : c(\mathbf{x}) = \bar{c}\}|}. \quad (1)$$

We define mAP as per the PASCAL VOC 2008 benchmark onward Everingham et al. (2008).[1]

DTD contains 10 preset splits into equally-sized training, validation and test subsets for easier algorithm comparison. Results on any of the tasks are repeated for each split and average accuracies are reported.

2.4 Attributes Versus Materials

As noted at the beginning of Sect. 2.1 and in Sharan et al. (2013), texture attributes and materials are correlated, but not equivalent. In this section we verify this quantitatively on the FMD data (Sharan et al. 2009). Specifically, we manually collected annotations for the 47 DTD attributes for the 1,000 images in the FMD dataset, which span ten different materials. Each of the 47 attributes was considered in turn, using a categorical random variable $C \in \{1, 2, \ldots, 10\}$ to denote the texture material and a binary variable $A \in \{0, 1\}$ to indicate whether the attribute applies to the texture or not. On average, the relative reduction in the entropy of the material variable $I(A, C)/H(C)$ given the attribute is of about 14 %; vice-versa, the relative reduction in the entropy of the attribute variable $I(A, C)/H(A)$ given the material is just 0.5 %. We conclude that knowing the material or attribute of a texture provides little information on the attribute or material, respectively. Note that *combinations* of attributes can predict materials much more reliably, although this is difficult to quantify from a small dataset.

---

[1] PASCAL VOC 2007 uses instead an interpolated version of mAP.

2.5 Related Work

This section relates DTD to the literature in texture understanding. Textures, due to their ubiquitousness and complementarity to other visual properties such as shape, have been studied in several contexts: texture perception (Adelson 2001; Amadasun and King 1989; Gårding 1992; Forsyth 2001), description (Ferrari and Zisserman 2007), material recognition (Leung and Malik 2001; Sharan et al. 2013; Schwartz and Nishino 2013; Varma and Zisserman 2005; Ojala et al. 2002; Varma and Zisserman 2003; Leung and Malik 2001, segmentation (Manjunath and Chellappa 1991; Jain and Farrokhnia 1991; Chaudhuri and Sarkar 1995; Manjunath and Chellappa 1991; Dunn et al. 1994), synthesis (Efros and Leung 1999; Wei and Levoy 2000; Portilla and Simoncelli 2000), and shape from texture (Gårding 1992; Forsyth 2001; Malik and Rosenholtz 1997). Most related to DTD is the work on texture recognition, summarized below as the recognition of perceptual properties (Sect. 2.5.1) and recognition of identities and materials (Sect. 2.5.2)

2.5.1 Recognition of Perceptual Properties

The study of perceptual properties of textures originated in computer vision as well as in cognitive sciences. Some of the earliest work on texture perception conducted by Julesz (1981) focussed on pre-attentive aspects of perception. It led to the concept of "textons," primitives such as line-terminators, crossings, intersections, etc., that are responsible for pre-attentive discrimination of textures. In computer vision, Tamura et al. (1978) identified six common directions of variability of images in the Broadatz dataset; coarse versus fine; high-contrast versus low-contrast; directional versus non-directional; linelike versus bloblike; regular versus irregular; and rough versus smooth. Similar perceptual attributes of texture (Amadasun and King 1989; Bajcsy 1973) have been found by other researchers.

Our work is motivated by that of Rao and Lohse (1996) and Bhushan et al. (1997). Their experiments suggest that there is a strong correlation between the structure of the lexical space and perceptual properties of texture. While they studied the psychological aspects of texture perception, the focus of this paper is the challenge of estimating such properties from images automatically. Their work Bhushan et al. (1997), in particular, identified a set of words sufficient to describe a wide variety of texture patterns; the same set of words was used to bootstrap DTD.

While recent work in computer vision has been focussed on texture identification and material recognition, notable contributions to the recognition of perceptual properties exist. Most of this work is part of the general research on *visual attributes* (Farhadi et al. 2009; Parikh and Grauman 2011; Patterson and Hays 2012; Bourdev et al. 2011; Kumar et al. 2011). Texture attributes have an important role in describing objects, particularly for those that are best characterized by a pattern, such as items of clothing and parts of animals such as birds. Notably, the first work on modern visual attributes by Ferrari and Zisserman (2007) focused on the recognition of a few perceptual properties of textures. Later work, such as Berg et al. (2010) that mined visual attributes from images on the Internet, also contain some attributes that describe textures. Nevertheless, so far the attributes of textures have been investigated only tangentially. DTD address the question of whether there exists a "universal" set of attributes that can describe a wide range of texture patterns, whether these can be reliably estimated from images, and for what tasks they are useful.

Datasets that focus on the recognition of subjective properties of textures are less common. One example is *Pertex* (Clarke et al. 2011), containing 300 texture images taken in a controlled setting (Lambertian renderings of 3D reconstructions of real materials) as well as a semantic similarity matrix obtained form human similarity judgments. The work most related to ours is probably the one of Matthews et al. (2013) that analyzed images in the Outex dataset (Ojala et al. 2002) using a subset of the texture attributes that we consider. DTD differs in scope (containing more attributes) and, especially, in the nature of the data (controlled vs uncontrolled conditions). In particular, working in uncontrolled conditions allows us to transfer the texture attributes to real-world applications, including material recognition in the wild and in clutter, as shown in the experiments.

2.5.2 Recognition of Texture Instances and Material Categories

Most of the recent work in texture recognition focuses on the recognition of texture instances and material categories, as reflected by the development of corresponding benchmarks (Fig. 4). The Brodatz (1966) catalogue was used in early works on textures to study the problem of identifying tex-
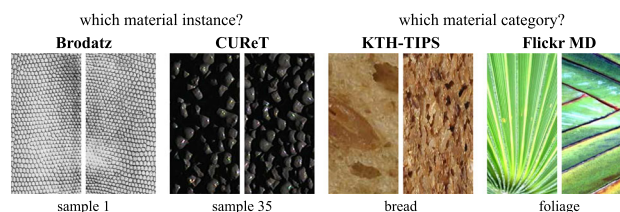

Fig. 4 Datasets such as Brodatz (1966) and CUReT (Dana et al. 1999) (*left*) addressed the problem of material instance identification and others such as. KTH-T2b (Hayman et al. 2004) and FMD (Sharan et al. 2009) (*right*) addressed the problem of material category recognition. Our DTD dataset addresses a very different problem: the one of describing a pattern using intuitive attributes (Fig. 1)

Table 1 Comparison of existing texture datasets, in terms of size, collection condition, nature of the classes to be recognized, and whether each class includes a single object/material instance or several instances of the same category

Dataset	Size			Condition			Content			(I)nstances /
	Images	Classes	Splits	Wild	Clutter	Controlled	Attributes	Materials	Objects	(C)ategories
Brodatz	999	111	–			X			X	I
CUReT	5612	61	10			X		X		I
UIUC	1000	25	10			X		X		I
UMD	1000	25	10	X					X	I
KTH	810	11	10			X		X		I
Outex	–	–	–			X		X	X	I
Drexel	∼40000	20	–			X		X		I
ALOT	25000	250	10			X		X		I
FMD	1000	10	14	X				X		C
KTH-T2b	4752	11				X		X		C
DTD	5640	47	10	X			X			C
OS	10422	22	1	X	X		X (+A)	X		C

Note that Outex is a meta-collection of textures spanning different datasets and problems

ture instances (e.g. matching half of the texture image given the other half). Others including *CUReT* (Dana et al. 1999), *UIUC* (Lazebnik et al. 2005), KTH-TIPS (Caputo et al. 2005; Hayman et al. 2004), *Outex* (Ojala et al. 2002), *Drexel Texture Database* (Oxholm et al. 2012), and *ALOT* (Burghouts and Geusebroek 2009) address the recognition of *specific instances of one or more materials*. *UMD* (Xu et al. 2009) is similar, but the imaged objects are not necessarily composed of a single material. As textures are imaged under variable truncation, viewpoint, and illumination, these datasets have stimulated the creation of texture representations that are invariant to viewpoint and illumination changes (Varma and Zisserman 2005; Ojala et al. 2002; Varma and Zisserman 2003; Leung and Malik 2001). Frequently, texture understanding is formulated as the problem of recognizing the material of an object rather than a particular texture instance (in this case any two slabs of marble would be considered equal). KTH-T2b (Mallikarjuna et al. 2006) is one of the first datasets to address this problem by grouping textures not only by the instance, but also by the type of materials (e.g. "wood").

However, these datasets make the simplifying assumption that textures fill images, and often, there is limited intra-class variability, due to *a single or limited number of instances*, captured under controlled scale, view-angle and illumination. Thus, they are not representative of the problem of recognizing materials in natural images, where textures appear under poor viewing conditions, low resolution, and in clutter. Addressing this limitation is the main goal of the *Flickr Material Database* (FMD) (Sharan et al. 2009). FMD samples just one viewpoint and illumination per object, but contains many different object instances grouped in several different mate-

rial classes. Sect. 3 will introduce datasets addressing the problem of clutter as well.

The performance of recognition algorithms on most of this data is close to perfect, with classification accuracies well above 95 %; KTH-T2b and FMD are an exception due to their increased complexity. A review of these datasets and classification methodologies is presented in Timofte and Van Gool (2012), who also propose a training-free framework to classify textures, significantly improving on other methods. Table 1 and Fig. 4 provides a summary of the nature and size of various texture datasets that are used in our experiments.

3 Recognizing Textures in Clutter

This section looks at the second contribution of the paper, namely studying the recognition of materials and describable textures attributes not only "in the wild," but also "in clutter". Even in datasets such as FMD and DTD, in fact, each texture instance fills the entire image, which doest not match most applications. This section removes this limitation and looks at the problem of recognizing textures imaged in the larger context of a complex natural scene, including the challenging task of automatically segmenting textured image regions.

Rather than collecting a new image dataset from scratch, our starting point is the excellent *open surfaces* (OS) dataset that was recently introduced by Bell et al. (2013). OS comprises 25,357 images, each containing a number of high-quality texture/material segments. Many of these segments are annotated with additional attributes such as the material, viewpoint, BRDF estimates, and object class. Experiments focus on the 58,928 segments that contain material anno-

tations. Since material classes are highly unbalanced, we consider only the materials that contain at least 400 examples. This results in 53,915 annotated material segments in 10,422 images spanning 23 different classes.[2] Images are split evenly into training, validation, and test subsets with 3474 images each. Segment sizes are highly variable, with half of them being relatively small, with an area smaller than 64×64 pixels. One issue with crowdsourced collection of segmentations is that not all the pixels in an image are labelled. This makes it difficult to define a complete background class. For our benchmark several less common materials (including for example segments that annotators could not assign to a material) were merged in an "other" class that acts as the background.

This benchmark is similar to the one concurrently proposed by Bell et al. (2015). However, in order to study perceptual properties as well as materials, we also augment the OS dataset with some of the describable attributes of Sect. 2. Since the OS segments do not trigger with sufficient frequency all the 47 attributes, the evaluation is restricted to eleven of them for which it was possible to identify at least 100 matching segments.[3] The attributes were manually labelled in the 53,915 segments retained for materials. We refer to this data as OSA.

3.1 Benchmark Tasks

As for DTD, the aim is to define standardized image understanding tasks to be used as public benchmarks. The complete list of images, segments, labels, and splits are publicly available at http://www.robots.ox.ac.uk/~vgg/data/wildtex/.

The benchmarks include two tasks on two complementary semantic domains. The first task is the *recognition of texture regions, given the region extent as ground truth information*. This task is instantiated for both material, denoted OS+R, and describable texture attributes, denoted OSA+R. Performance in OS+R is measured in term of classification accuracy and mAP, using the same definition (1) where images are replaced by image regions. Performance in OSA+R uses instead mAP due to the possibility of multiple labels.

The second task is the *segmentation and recognition of texture regions*, which we also instantiate for materials (OS) and describable texture attributes (OSA). Since not all image

pixels are labelled in the ground truth, the performance of a predictor $\hat{c}$ is measured in term of per-pixel classification accuracy, pp-acc($\hat{c}$). This is computed using the same formula as (1) with two modification: first, the images $\mathbf{x}$ are replaced by pixels $\mathbf{p}$ (extracted from all images in the dataset); second, the ground truth label $c(\mathbf{p})$ of a pixel may take an additional value 0 to denote pixels that are not labelled in the ground truth (the effect is to ignore them in the computation of accuracy).

In the case of OSA, the per-pixel accuracy is modified such that a class prediction is considered correct if it belongs to any of the ground-truth pixel labels. Furthermore, accuracy is not normalized per class as this is ill-defined, but by the total number of pixels:

$$\text{acc-osa}(\hat{c}) = \frac{|\{\mathbf{p} : \hat{c}(\mathbf{p}) \in c(\mathbf{p})\}|}{|\{\mathbf{p} : c(\mathbf{p}) \neq \phi\}|}. \tag{2}$$

where $c(\mathbf{p})$ is the set of possible labels of pixel $\mathbf{p}$ and ϕ denotes the empty set.

4 Texture Representations

Having presented our contributions to framing the problem of texture description, we now turn to our technical advances towards addressing the resulting problems. We start by revisiting the concept of texture representation and studies how it relates to modern image descriptors based on CNNs. In general, a visual representation is a map that takes an image $\mathbf{x}$ to a vector $\phi(\mathbf{x}) \in \mathbb{R}^d$ that facilitates understanding the image content. Understanding is often achieved by learning a linear predictor $\langle \phi(\mathbf{x}), \mathbf{w} \rangle$ scoring the strength of association between the image and a particular concept, such as an object category.

Among image representations, this paper is particularly interested in the class of *texture representations* pioneered by the works of Mallat (1989), Malik and Perona (1990), Bovik et al. (1990), and Leung and Malik (2001). Textures encompass a large diversity of visual patterns, from regular repetitions such as wallpapers, to stochastic processes such as fur, to intermediate cases such as pebbles. Distortions due to viewpoint and other imaging factors further complicate modeling textures. However, one can usually assume that, given a particular texture, appearance variations are statistically independent in the long range and can therefore be eliminated by averaging local image statistics over a sufficiently large texture sample. Hence, the defining characteristic of texture representations is to pool information extracted locally and uniformly from the image, by means of local descriptors, in an orderless manner.

The importance of texture representations is in the fact that they were found to be applicable well beyond textures. For

---

[2] The classes and corresponding number of example segments are: brick (610), cardboard (423), carpet/rug (1975), ceramic (1643), concrete (567), fabric/cloth (7484), food (1461), glass (4571), granite/marble (1596), hair (443), other (2035), laminate (510), leather (957), metal (4941), painted (7870), paper/tissue (1226), plastic/clear (586), plastic/opaque (1800), stone (417), tile (3085), wallpaper (483), wood (9232).

[3] These are: banded, blotchy, checkered, flecked, gauzy, grid, marbled, paisley, pleated, stratified, wrinkled.

example, until recently many of the best object categorization methods in challenges such as PASCAL VOC (Everingham et al. 2007) and ImageNet ILSVRC (Deng et al. 2009) were based on variants of texture representations, developed specifically for objects. One of the contributions of this work is to show that these object-optimized texture representations are in fact optimal for a large number of texture-specific problems too (Sect. 6.1.3).

More recently, texture representations have been significantly outperformed by *Convolutional Neural Networks* (CNNs) in object categorization (Krizhevsky et al. 2012), detection (Girshick et al. 2014), segmentation (Hariharan et al. 2014), and in fact in almost all domains of image understanding. Key to the success of CNNs is their ability to leverage large *labelled* datasets to learn high-quality features. Importantly, CNN features pre-trained on very large datasets were found to transfer to many other domains with a relatively modest adaptation effort (Jia 2013; Oquab et al. 2014; Razavin et al. 2014; Chatfield et al. 2014; Girshick et al. 2014). Hence, CNNs provide general-purpose image descriptors.

While CNNs generally outperform classical texture representations, it is interesting to ask what is the relation between these two methods and whether they can be fruitfully hybridized. Standard CNN-based methods such as Jia (2013), Oquab et al. (2014), Razavin et al. (2014), Chatfield et al. (2014), and Girshick et al. (2014) can be interpreted as extracting local image descriptors (performed by the the so called "convolutional layers") followed by pooling such features in a global image representation (performed by the "Fully-Connected (FC) layers"). Here we will show that replacing FC pooling with one of the many pooling mechanisms developed in texture representations has several advantages: (i) a much faster computation of the representation for image subregions accelerating applications such as detection and segmentation (Girshick et al. 2014; He et al. 2014; Gong et al. 2014), (ii) a significantly superior recognition accuracy in several application domains and (iii) the ability of achieving this superior performance without fine-tuning CNNs by implicitly reducing the domain shift problem.

In order to systematically study variants of texture representations $\phi = \phi_e \circ \phi_f$, we break them into local descriptor extraction ϕ_f followed by descriptor pooling ϕ_e. In this manner, different combinations of each component can be evaluated. Common local descriptors include linear filters, local image patches, local binary patterns, densely-extracted SIFT features, and many others. Since local descriptors are extracted uniformly from the image, they can be seen as banks of (non-linear) filters; we therefore refer to them as *filter banks* in honor of the pioneering works of Mallat (1989), Bovik et al. (1990), Freeman and Adelson (1991), Leung and Malik (2001) and others where descriptors were the output

of actual linear filters. Pooling methods include bag-of-visual-words, variants using soft-assignment, or extracting higher-order statistics as in the Fisher vector. Since these methods encode the information contained in the local descriptors in a single vector, we refer to them as *pooling encoders*.

Sects. 4.1 and 4.2 discuss filter banks and pooling encoders in detail.

4.1 Local Image Descriptors

There is a vast choice of local image descriptors in texture representations. Traditionally, these features were hand-crafted , but with the latest generation of deep learning methods it is now customary to learn them from data (although often in an implicit form). Representative examples of these two families of local features are discussed in Sects. 4.1.1 and 4.1.2, respectively.

4.1.1 Hand-Crafted Local Descriptors

Some of the earliest local image descriptors were developed as linear filter banks in texture recognition. As an evolution of earlier texture filters (Bovik et al. 1990; Malik and Perona 1990), the filter bank of **Leung Malik** (LM) Leung and Malik (2001) includes 48 filters matching bars, edges and spots, at various scales and orientations. These filters are first and second derivatives of Gaussians at 6 orientations and 3 scales (36), 8 Laplacian of Gaussian (LOG) filters, and 4 Gaussians. Combinations of the filter responses, identified by vector quantisation (Sect. 4.2.1), were used as the computational basis of the "textons" proposed by Julesz and Bergen (1983). The filter bank **MR8** of Varma and Zisserman (2003) and Geusebroek et al. (2003) consists instead of 38 filters, similar to LM. For two of the oriented filters, only the maximum response across the scales is recorded, reducing the number of responses to 8 (3 scales for two oriented filters, and two isotropic – Gaussian and Laplacian of Gaussian).

The importance of using linear filters as local features was later questioned by Varma and Zisserman (2003). The **VZ** descriptors are in fact small image patches which, remarkably, were shown to outperform LM and MR8 on earlier texture benchmarks such as CuRET. However, as will be demonstrated in the experiments, trivial local descriptors are not competitive in harder tasks.

Another early local image descriptor are the **Local Binary Patterns** (LBP) of Ojala et al. (1996) and Ojala et al. (2002), a special case of the texture units of Wang and He (1990). A LBP $\mathbf{d}_i = (b_1, \ldots, b_m)$ computed a pixel $\mathbf{p}_0$ is the sequence of bits $b_j = [\mathbf{x}(\mathbf{p}_i) > \mathbf{x}(\mathbf{p}_j)]$ comparing the intensity $\mathbf{x}(\mathbf{p}_i)$ of the central pixel to the one of m neighbors $\mathbf{p}_j$ (usually 8 in a circle). LBPs have specialized quantization schemes; the most common one maps the bit string $\mathbf{d}_i$ to one of a number of

uniform patterns (Ojala et al. 2002). The quantized LBPs can be averaged over the image to build a histogram; alternatively, such histograms can be computed for small image patches and used in turn as local image descriptors.

In the context of object recognition, the best known local descriptor is undoubtedly Lowe's **SIFT** (Lowe 1999). SIFT is the histogram of the occurrences of image gradients quantized with respect to their location within a patch as well to their orientation. While SIFT was originally introduced to match object instances, it was later applied to an impressive diversity of tasks, from object categorization to semantic segmentation and face recognition.

4.1.2 Learned Local Descriptors

Handcrafted image descriptors are nowadays outperformed by features learned using the latest generation of deep CNNs (Krizhevsky et al. 2012). A CNN can be seen as a composition $\phi_K \circ \cdots \circ \phi_2 \circ \phi_1$ of K functions or *layers*. The output of each layer $\mathbf{x}_k = (\phi_k \circ \cdots \circ \phi_2 \circ \phi_1)(\mathbf{x})$ is a *descriptor field* $\mathbf{x}_k \in \mathbb{R}^{W_k \times H_k \times D_k}$, where W_k and H_k are the width and height of the field and D_k is the number of feature channels. By collecting the D_k responses at a certain spatial location, one obtains a D_k dimensional descriptor vector. The network is called convolutional if all the layers are implemented as (non-linear) filters, in the sense that they act locally and uniformly on their input. If this is the case, since compositions of filters are filters, the feature field $\mathbf{x}_k$ is the result of applying a non-linear filter bank to the image $\mathbf{x}$.

As computation progresses, the resolution of the descriptor fields decreases whereas the number of feature channels increases. Often, the last several layers ϕ_k of a CNN are called "fully connected" because, if seen as filters, their support is the same as the size of the input field $\mathbf{x}_{k-1}$ and therefore lack locality. By contrast, earlier layers that act locally will be referred to as "convolutional". If there are C convolutional layers, the CNN $\phi = \phi_e \circ \phi_f$ can be decomposed into a filter bank (local descriptors) $\phi_f = \phi_C \circ \cdots \circ \phi_1$ followed by a pooling encoder $\phi_e = \phi_K \circ \cdots \circ \phi_{C+1}$.

4.2 Pooling Encoders

A pooling encoder takes as input the local descriptors extracted from an image $\mathbf{x}$ and produces as output a single feature vector $\phi(\mathbf{x})$, suitable for tasks such as classification with an SVM. A first important differentiating factor between encoders is whether they discard the spatial configuration of input features (orderless pooling; Sect. 4.2.1) or whether they retain it (order-sensitive pooling; Sect. 4.2.2). A detail of practical importance, furthermore, is the type of post-processing applied to the pooled vectors (post-processing; Sect. 4.2.3).

4.2.1 Orderless Pooling Encoders

An *orderless pooling encoder* ϕ_e maps a sequence $\mathcal{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_n), \mathbf{f}_i \in \mathbb{R}^D$ of local image descriptors to a feature vector $\phi_e(\mathcal{F}) \in \mathbb{R}^d$. The encoder is orderless in the sense that the function ϕ_e is invariant to permutations of the input $\mathcal{F}$.[4] Furthermore, the encoder can be applied to any number of features; for example, the encoder can be applied to the sub-sequence $\mathcal{F}' \subset \mathcal{F}$ of local descriptors contained in a target image region without recomputing the local descriptors themselves.

All common orderless encoders are obtained by applying a non-linear *descriptor encoder* $\eta(\mathbf{f}_i) \in \mathbb{R}^d$ to individual local descriptors and then aggregating the result by using a commutative operator such as average or max. For example, average-pooling yields $\bar{\phi}_e(\mathcal{F}) = \frac{1}{n} \sum_{i=1}^{n} \eta(\mathbf{f}_i)$. The pooled vector $\bar{\phi}_e(\mathcal{F})$ is post-processed to obtain the final representation $\phi_e(\mathcal{F})$ as discussed later.

The best-known orderless encoder is the **Bag of Visual Words** (BoVW). This encoder starts by vector-quantizing (VQ) the local features $\mathbf{f}_i \in \mathbb{R}^D$ by assigning them to their closest *visual word* in a dictionary $C = [\mathbf{c}_1 \ldots \mathbf{c}_d] \in \mathbb{R}^{D \times d}$ of d elements. Visual words can be thought of as "prototype features" and are obtained during training by clustering example local features. The descriptor encoder $\eta_1(\mathbf{f}_i)$ is the one-hot vector indicating the visual word corresponding to $\mathbf{f}_i$ and average-pooling these one-hot vectors yields the histogram of visual words occurrences. BoVW was introduced in the work of Leung and Malik (2001) to characterize the distribution of textons, defined as configuration of local filter responses, and then ported to object instance and category understanding by Sivic and Zisserman (2003) and Csurka et al. (2004) respectively. It was then extended in several ways as described below.

The kernel codebook encoder (Philbin et al. 2008) assigns each local feature to several visual words, weighted by a degree of membership: $[\eta_{KC}(\mathbf{f}_i)]_j \propto \exp\left(-\lambda \|\mathbf{f}_i - \mathbf{c}_j\|^2\right)$, where λ is a parameter controlling the locality of the assignment. The descriptor code $\eta_{KC}(\mathbf{f}_i)$ is L^1 normalized before aggregation, such that $\|\eta_{KC}(\mathbf{f}_i)\|_1 = 1$. Several related methods used concepts from *sparse coding* to define the local descriptor encoder (Zhou et al. 2010; Liu et al. 2011). **Locality constrained linear coding** (LLC) (Wang et al. 2010), in particular, extends soft assignment by making the assignments reconstructive, local, and sparse: the descriptor encoder $\eta_{LLC}(\mathbf{f}_i) \in \mathbb{R}_+^d, \|\eta_{LLC}(\mathbf{f}_i)\|_1 = 1, \|\eta_{LLC}(\mathbf{f}_i)\|_0 \leq r$ is computed such that $\mathbf{f}_i \approx C\eta_{LLC}(\mathbf{f}_i)$ while allowing only the $r \ll d$ visual words closer to $\mathbf{f}_i$ to have a non-zero coefficient.

---

[4] Note that $\mathcal{F}$ cannot be represented as a set as encoders are generally sensitive to repetitions of feature descriptors. It could be defined as a multiset or, as done here, as a sequence $\mathcal{F}$.

In the **Vector of locally-aggregated descriptors** (VLAD) (Jégou et al. 2010) the descriptor encoder is richer. Local image descriptors are first assigned to their nearest neighbor visual word in a dictionary of K elements like in BoVW; then the descriptor encoder is given by $\eta_{\text{VLAD}}(\mathbf{f}_i) = (\mathbf{f}_i - C\eta_1(\mathbf{f}_i)) \otimes \eta_1(\mathbf{f}_i)$, where $\otimes$ is the Kronecker product. Intuitively, this subtracts from $\mathbf{f}_i$ the corresponding visual word $C\eta_1(\mathbf{f}_i)$ and then copies the difference into one of K possible subvectors, one for each visual word. Hence average-pooling $\eta_{\text{VLAD}}(\mathbf{f}_i)$ accumulates first-order descriptor statistics instead of simple occurrences as in BoVW.

VLAD can be seen as a variant of the **Fisher vector** (FV) (Perronnin and Dance 2007). The FV differs from VLAD as follows. First, the quantizer is not K-means but a Gaussian Mixture Model (GMM) with components (π_k, μ_k, Σ_k), $k = 1, \ldots, K$, where $\pi_k \in \mathbb{R}$ is the prior probability of the component, $\mu_k \in \mathbb{R}^D$ the Gaussian mean and $\Sigma_k \in \mathbb{R}^{D \times D}$ the Gaussian covariance (assumed diagonal). Second, hard-assignments $\eta_1(\mathbf{f}_i)$ are replaced by soft-assignments $\eta_{\text{GMM}}(\mathbf{f}_i)$ given by the posterior probability of each GMM component. Third, the FV descriptor encoder $\eta_{\text{FV}}(\mathbf{f}_i)$ includes both first $\Sigma_k^{-\frac{1}{2}}(\mathbf{f}_i - \mu_k)$ and second order $\Sigma_k^{-1}(\mathbf{f}_i - \mu_k) \odot (\mathbf{f}_i - \mu_k) - \mathbf{1}$ statistics, weighted by $\eta_{\text{GMM}}(\mathbf{f}_i)$ [see Perronnin and Dance (2007), Perronnin et al. (2010) and Chatfield et al. (2011) for details]. Hence, average pooling $\eta_{\text{FV}}(\mathbf{f}_i)$ accumulates both first and second order statistics of the local image descriptors.

All the encoders discussed above use average pooling, except LLC that uses max pooling.

4.2.2 Order-Sensitive Pooling Encoders

An *order-sensitive encoder* differs from an orderless encoder in that the map $\phi_e(\mathcal{F})$ is not invariant to permutation of the input $\mathcal{F}$. Such an encoder can therefore reflect the layout of the local image desctiptors, which may be ineffective or even counter-productive in texture recognition, but is usually helpful in the recognition of objects, scenes, and others.

The most common order-sensitive encoder method is the **spatial pyramid pooling** (SPP) of Lazebnik et al. (2006). SSP transforms any orderless encoder into one with (weak) spatial sensitivity by dividing the image in subregions, computing any encoder for each subregion, and stacking the results. This encoder is only be sensitive to reassignments of the local descriptors to different subregions.

The **fully-connected layers** (FC) in a CNN also form an order-sensitive encoder. Compared to the encoders seen above, FC are pre-trained discriminatively, which can be either an advantage or disadvantage, depending on whether the information that they captured can be transferred to the domain of interest. FC poolers are much less flexible than the encoders seen above as they work only with a particular type of local descriptors, namely the corresponding CNN convolutional layers. Furthermore, a standard FC pooler can only operate on a well defined layout of local descriptors (e.g. 6×6), which in turn means that the image needs to be resized to a standard size before the FC encoder can be evaluated. This is particularly expensive when, as in object detection or image segmentation, many image subregions must be considered.

4.2.3 Post-processing

The vector $\mathbf{y} = \phi_e(\mathcal{F})$ obtained by pooling local image descriptors is usually post-processed before being used in a classifier. In the simplest case, this amounts to performing L^2 normalization $\phi_e(\mathcal{F}) = \mathbf{y}/\|\mathbf{y}\|_2$. However, this is usually preceded by a non-linear transformation $\phi_K(\mathbf{y})$ which is best understood in term of kernels. A *kernel* $K(\mathbf{y}', \mathbf{y}'')$ specifies a notion of *similarity* between data points $\mathbf{y}'$ and $\mathbf{y}''$. If K is a positive semidefinite function, then it can always be rewritten as the inner product $\langle \phi_K(\mathbf{y}'), \phi_K(\mathbf{y}'') \rangle$ where ϕ_K is a suitable pre-processing function called a *kernel embedding* (Maji et al. 2008; Vedaldi and Zisserman 2010). Typical kernels include the linear, Hellinger's, additive-χ^2, and exponential-χ^2 ones, given respectively by:

$$\langle \mathbf{y}', \mathbf{y}'' \rangle, \qquad \sum_{i=1}^{d} \sqrt{y_i' y_i''},$$

$$\sum_{i=1}^{d} \frac{2y_i' y_i''}{y_i' + y_i''}, \qquad \exp\left[-\lambda \sum_{i=1}^{d} \frac{(y_i' - y_i'')^2}{y_i' + y_i''} \right].$$

In practice, the kernel embedding ϕ_K can be computed easily only in a few cases, including the linear kernel (ϕ_K is the identity) and Hellinger's kernel (for each scalar component, $\phi_{\text{Hell.}}(y) = \sqrt{y}$). In the latter case, if y can take negative values, then the embedding is extended to the so called *signed square rooting*[5] $\phi_{\text{Hell.}}(y) = \text{sign}(y)\sqrt{|y|}$.

Even if ϕ_K is not explicitly computed, any kernel can be used to learn a classifier such as an SVM (kernel trick). In this case, L^2 normalizing the kernel embedding $\phi_K(\mathbf{y})$ amounts to normalizing the kernel as

$$K'(\mathbf{y}, \mathbf{y}'') = \frac{K(\mathbf{y}', \mathbf{y}'')}{\sqrt{K(\mathbf{y}', \mathbf{y}') K(\mathbf{y}'', \mathbf{y}'')}}.$$

All the pooling encoders discussed above are usually followed by post-processing. In particular, the *Improved Fisher Vector* (IFV) (Perronnin et al. 2010) prescribes the use of the signed-square root embedding followed by L^2 normalization. VLAD has several standard variants that differ in

---

[5] This extension generalizes to all homogeneous kernels, including for example χ^2 (Vedaldi and Zisserman 2010).

the post-processing; here we use the one that L^2 normalizes the individual VLAD subvectors (one for each visual word) before L^2 normalizing the whole vector (Arandjelovic and Zisserman 2012).

5 Plan of Experiments and Highlights

The next several pages contain an extensive set of experimental results. This section provides a guide to these experiments and summarizes the main findings.

The goal of the first block of experiments (Sect. 6.1) is to determine which representations work bests on different problems such as texture attribute, texture material, object, and scene recognition. The main findings are:

- Orderless pooling of SIFT features (e.g. FV-SIFT) performs better than specialized texture descriptors in many texture recognition problems; performance is further improved by switching from SIFT to CNN local descriptors (FV-CNN; Sect. 6.1.3).
- Orderless pooling of CNN descriptors using the Fisher Vector (FV-CNN) is often significantly superior than fully-connected pooling of the same descriptors (FC-CNN) in texture, scene, and object recognition (Sect. 6.1.4). This difference is more marked for deeper CNN architectures (Sect. 6.1.5) and can be partially explained by the ability of FV pooling to overfit less and to easily integrate information at multiple image scales (Sect. 6.1.6).
- FV-CNN descriptors can be compressed to the same dimensionality of FC-CNN descriptors while preserving accuracy (Sect. 6.1.7).

Having determined good representations in Sect. 6.1, the second block of experiments (Sect. 6.2) compares them to the state of the art in texture, object, and scene recognition. The main findings are:

- In texture recognition in the wild, for both materials (FMD) and attributes (DTD), CNN-based descriptors substantially outperform existing methods. Depending on the dataset, FV pooling is a little or substantially better than FC pooling of CNN descriptors (Sect. 6.2.1.4). When textures are extracted from a larger cluttered scene (instead of filling the whole image), the difference between FV and FC pooling increases (Sect. 6.2.1.5).
- In coarse object recognition (PASCAL VOC), fine-grained object recognition (CUB-200), scene recognition (MIT Indoor), and recognition of things & stuff (MSRC) fine-grained, the FV-CNN representation achieves results that are close and sometimes superior to the state of

the art, while using a simple and fully generic pipeline (Sect. 6.2.3).
- FV-CNN appears to be particularly effective in domain transfer. Sect. 6.2.3 shows in fact that FV pooling compensates for the domain gap caused by training a CNN on two very different domains, namely scene and object recognition.

Having addressed image classification in Sects. 6.1 and 6.2, The third block of experiments (Sect. 7) compare representations on semantic segmentation. It shows that FV pooling of CNN descriptors can be combined with a region proposal generator to obtain high-quality segmentation of materials in the OS and MSRC data. For example, combined with a post-processing step using a CRF, FV-VGG-VD surpasses the state-of-the-art on the latter dataset. It is also shown that, differently from FV-CNN, FC-CNN is too slow to be practical in this scenario.

6 Experiments on Semantic Recognition

So far the paper has introduced novel problems in texture understanding as well as a number of old and new texture representations. The goal of this section is to determine, through extensive experiments, what representations work best for which problem.

Representations are labelled as pairs X-Y, where X is a pooling encoder and Y a local descriptor. For example, FV-SIFT denotes the Fisher vector encoder applied to densely extracted SIFT descriptors, whereas BoVW-CNN denotes the bag-of-visual-words encoder applied on top of CNN convolutional descriptors. Note in particular that the CNN-based image representations as commonly extracted in the literature Jia (2013), Razavin et al. (2014), and Chatfield et al. (2014) implicitly use CNN-based descriptors and the FC pooler, and therefore are denoted here as FC-CNN.

6.1 Local Image Descriptors and Encoders Evaluation

This section compares different local image descriptors and pooling encoders (Sect. 6.1.1) on selected representative tasks in texture recognition, object recognition, and scene recognition (Sect. 6.1.2). In particular, Sect. 6.1.3 compares different local descriptors, Sect. 6.1.4 different pooling encoders, and Sect. 6.1.5 additional variants of the CNN-based descriptors.

6.1.1 General Experimental Setup

The experiments are centered around two types of local descriptors. The first type are SIFT descriptors extracted densely from the image (denoted *DSIFT*). SIFT descriptors

are sampled with a step of two pixels and the support of the descriptor is scaled such that a SIFT spatial bin has size 8×8 pixels. Since there are 4×4 spatial bins, the support or "receptive field" of each DSIFT descriptor is 40×40 pixels, (including a border of half a bin due to bilinear interpolation). Descriptors are 128-dimensional (Lowe 1999), but their dimensionality is further reduced to 80 using PCA, in all experiments. Besides improving the classification accuracy, this significantly reduces the size of the Fisher Vector and VLAD encodings.

The second type of local image descriptors are deep convolutional features (denoted *CNN*) extracted from the convolutional layers of CNNs pre-trained on ImageNet ILSVRC data. Most experiments build on the VGG-M model of Chatfield et al. (2014) as this network performs better than standard networks such as the Caffe reference model (Jia 2013) and AlexNet (Krizhevsky et al. 2012) while having a similar computational cost. The VGG-M convolutional features are extracted as the output of the last convolutional layer, directly from the linear filters excluding ReLU and max pooling, which yields a field of 512-dimensional descriptor vectors. In addition to VGG-M, experiments consider the recent VGG-VD (very deep with 19 layers) model of Simonyan and Zisserman (2014). The receptive field of CNN descriptors is much larger compared to SIFT: 139×139 pixels for VGG-M and 252×252 for VGG-VD.

When combined with a pooling encoder, local descriptors are extracted at multiple scales, obtained by rescaling the image by factors 2^s, $s = -3, -2.5, \ldots, 1.5$ (but, for efficiency, discarding scales that would make the image larger than 1024^2 pixels).

The dimensionality of the final representation strongly depends on the encoder type and parameters. For K visual words, BoVW and LLC have K dimensions, VLAD has KD and FV $2KD$, where D is the dimension of the local descriptors. For the FC encoder, the dimensionality is fixed by the CNN architecture; here the representation is extracted from the penultimate FC layer (before the final classification layer) of the CNNs and happens to have 4096 dimensions for all the CNNs considered. In practice, dimensions vary widely, with BoVW, LLC, and FC having a comparable dimensionality, and VLAD and FV a much higher one. For example, FV-CNN has $\simeq 64 \cdot 10^3$ dimensions with $K = 64$ Gaussian mixture components, versus the 4096 of FC, BoVW, and LLC (when used with $K = 4096$ visual words). In practice, however, dimensions are hardly comparable as VLAD and FV vectors are usually highly compressible (Parkhi et al. 2014). We verified that by using PCA to reduce FV to 4096 dimensions and observing only a marginal reduction in classification performance in the PASCAL VOC object recognition task, as described below.

Unless otherwise specified, learning uses a standard non-linear SVM solver. Initially, cross-validation was used to select the parameter C of the SVM in the range {0.1, 1, 10, 100}; however, after noting that performance was nearly identical in this range (probably due to the data normalization), C was simply set to the constant 1. Instead, it was found that recalibrating the SVM scores for each class improves classification accuracy (but of course not mAP). Recalibration is obtained by changing the SVM bias and rescaling the SVM weight vector in such a way that the median scores of the negative and positive training samples for each class are mapped respectively to the values -1 and 1.

All the experiments in the paper use the VLFeat library (Vedaldi and Fulkerson 2010) for the computation of SIFT features and the pooling embedding (BoVW, VLAD, FV). The MatConvNet (Vedaldi and Lenc 2014) library is used instead for all the experiments involving CNNs. Further details specific to the setup of each experiment are given below as needed.

6.1.2 Datasets and Evaluation Measures

The evaluation is performed on a diversity of tasks: the new describable attribute and material recognition benchmarks in DTD and OpenSurfaces, existing ones in FMD and KTH-T2b, object recognition in PASCAL VOC 2007, and scene recognition in MIT Indoor. All experiments follow standard evaluation protocols for each dataset, as detailed below.

DTD (Sect. 2) contains 47 texture classes, one per visual attribute, containing 120 images each. Images are equally split into train, test and validation, and include experiments on the prediction of "key attributes" as well as "joint attributes", as as defined in Sect. 2.1, and reports accuracy averaged over the 10 default splits provided with the datasets. **OpenSurfaces** (Bell et al. 2013) is used in the setup described in Sect. 3 and contains 25,357 images, out of which we selected 10,422 images, spanning across 21 categories. When segments are provided, the dataset is referred to as OS+R, and recognition accuracy is reported on a per-segment basis. We also annotated the segments with the attributes from DTD, and called this subset OSA (and OSA+R for the setup when segments are provided). For the recognition task on OSA+R we report mean average precision, as this is a multi-label dataset.

FMD (Sharan et al. 2009) consists of 1,000 images with 100 for each of ten material categories. The standard evaluation protocol of Sharan et al. (2009) uses 50 images per class for training and the remaining 50 for testing, and reports classification accuracy averaged over 14 splits. **KTH-T2b** [65] contains 4,752 images, grouped into 11 material categories. For each material category, images of four samples were captured under various conditions, resulting in 108 images per sample. Following the standard procedure (Caputo et al. 2005; Timofte and Van Gool 2012), images of one material sample are used to train the model, and the other three

samples for evaluating it, resulting in four possible splits of the data, for which average per-class classification accuracy is reported. **MIT Indoor Scenes** (Quattoni and Torralba 2009) contains 6,700 images divided in 67 scene categories. There is one split of the data into train (80 %) and test (20 %), provided with the dataset, and the evaluation metric is average per-class classification accuracy. **PASCAL VOC 2007** (Everingham et al. 2007) contains 9963 images split across 20 object categories. The dataset provides a standard split in training, validation and test data. Performance is reported in term of mean average precision (mAP) computed using the TRECVID 11-point interpolation scheme (Everingham et al. 2007).[6]

6.1.3 Local Image Descriptors and Kernels Comparison

The goal of this section is to establish which local image descriptors work best in a texture representation. The question is relevant because: (i) while SIFT is the de-facto standard handcrafted-feature in object and scene recognition, most authors use specialized descriptors for texture recognition and (ii) learned convolutional features in CNNs have not yet been compared when used as local descriptors (instead, they have been compared to classical image representations when used in combination with their FC layers).

The experiments are carried on the the task of recognizing describable texture attributes in DTD (Sect. 2) using the BoVW encoder. As a byproduct, the experiments determine the relative difficulty of recognizing the different 47 perceptual attributes in DTD.

6.1.3.1 Experimental Setup The following local image descriptors are compared: the linear filter banks of *Leung and Malik* (LM) (Leung and Malik 2001) (48D descriptors) and *MR8* (8D descriptors) (Varma and Zisserman 2005; Geusebroek et al. 2003), the 3×3 and 7×7 raw image patches of Varma and Zisserman (2003) (respectively 9D and 49D), the *local binary patterns* (LBP) of Ojala et al. (2002) (58D), SIFT (128D), and CNN features extracted from *VGG-M* and *VGG-VD* (512D).

After the BoVW representation is extracted, it is used to train a 1-vs-all SVM using the different kernels discussed in Sect. 4.2.3: linear, Hellinger, additive-χ^2, and exponential-χ^2. Kernels are normalized as described before. The exponential-χ^2 kernel requires choosing the parameter λ; this is set as the reciprocal of the mean of the χ^2 distance matrix of the training BoVW vectors. Before computing the exponential-χ^2 kernel, furthermore, BoVW vectors are L^1 normalized. An important parameter in BoVW is the number of visual words selected. K was varied in the range of

256, 512, 1024, 2048, 4096 and performance evaluated on a validation set. Regardless of the local feature and embedding, performance was found to increase with K and to saturate around $K = 4096$ (although the relative benefit of increasing K was larger for features such as SIFT and CNNs). Therefore K was set to this value in all experiments.

6.1.3.2 Analysis Table 2 reports the classification accuracy for 47 1-vs-all SVM attribute classifiers, computed as (1). As often found in the literature, the best kernel was found to be exponential-χ^2, followed by additive-χ^2, Hellinger's, and linear kernels. Among the hand-crafted descriptors, dense SIFT significantly outperforms the best specialized texture descriptor on the DTD data (52.3 % for BoVW-exp-χ^2-SIFT vs 44 % for BoVW-exp-χ^2-LM). CNN local descriptors handily outperform handcrafted features by a 10–15 % recognition accuracy margin. It is also interesting to note that the choice of kernel function has a much stronger effect for image patches and linear filters (e.g. accuracy nearly doubles moving from BoVW-linear-patches to BoVW-exp-χ^2-patches) and an almost negligible effect for the much stronger CNN features.

Figure 5 reports the classification accuracy for each attribute in DTD for the BoVW-SIFT, BoVW-VGG-M, and BoVW-VGG-VD descriptors and the additive-χ^2 kernel. As it may be expected, concepts such as *chequered*, *waffled*, *knitted*, *paisley* achieve nearly perfect classification, while others such as *blotchy*, *smeared* or *stained* are far harder.

6.1.3.3 Conclusions The conclusions are that (i) SIFT descriptors outperform significantly texture-specific descriptors such as linear filter banks, patches, and LBP on this texture recognition task, and that (ii) learned convolutional local descriptors significantly surpass SIFT.

6.1.4 Pooling Encoders

The previous section established the primacy of SIFT and CNN local image descriptors on alternatives. The goal of this section is to determine which pooling encoders (Sect. 4.2) work best with these descriptors, comparing the orderless BoVW, LLC, VLAD, FV encoders and the order-sensitive FC encoder. The latter, in particular, reproduces the CNN transfer learning setting commonly found in the literature where CNN features are extracted in correspondence to the FC layers of a network.

6.1.4.1 Experimental Setup The experimental setup is similar to the previous experiment: the same SIFT and CNN VGG-M descriptors are used; BoVW is used in combination with the Hellinger kernel (the exponential variant is slightly better, but much more expensive); the same $K = 4096$ codebook size is used with LLC. VLAD and FV use a

---

[6] The procedure for computing the AP was changed in later versions of the benchmark.

Table 2 Comparison of local features and kernels on the DTD data

Local descr.	Kernel			
	Linear	Hellinger	add-χ^2	exp-χ^2
MR8	20.8 ± 0.9	26.2 ± 0.8	29.7 ± 0.9	34.3 ± 1.1
LM	26.7 ± 0.9	34.8 ± 1.2	39.5 ± 1.4	44.0 ± 1.4
Patch$_{3 \times 3}$	15.9 ± 0.5	24.4 ± 0.7	27.8 ± 0.8	30.9 ± 0.7
Patch$_{7 \times 7}$	20.7 ± 0.8	30.6 ± 1.0	34.8 ± 1.0	37.9 ± 0.9
LBP^u	8.5 ± 0.4	9.3 ± 0.5	12.5 ± 0.4	19.4 ± 0.7
LBP-VQ	26.2 ± 0.8	28.8 ± 0.9	32.7 ± 1.0	36.1 ± 1.3
SIFT	$\mathbf{45.2 \pm 1.0}$	$\mathbf{49.1 \pm 1.1}$	$\mathbf{50.9 \pm 1.0}$	$\mathbf{52.3 \pm 1.2}$
Conv VGG-M	$\mathbf{55.9 \pm 1.3}$	$\mathbf{61.7 \pm 0.9}$	$\mathbf{61.9 \pm 1.0}$	$\mathbf{61.2 \pm 1.0}$
Conv VGG-VD	$\mathbf{64.1 \pm 1.3}$	$\mathbf{68.8 \pm 1.3}$	$\mathbf{69.0 \pm 0.9}$	$\mathbf{68.8 \pm 0.9}$

The table reports classification accuracy, averaged over the predefined ten splits, provided with the dataset We marked in bold the best performing descriptors, SIFT and convolutional features, which we will cover in the following experiments and discussions
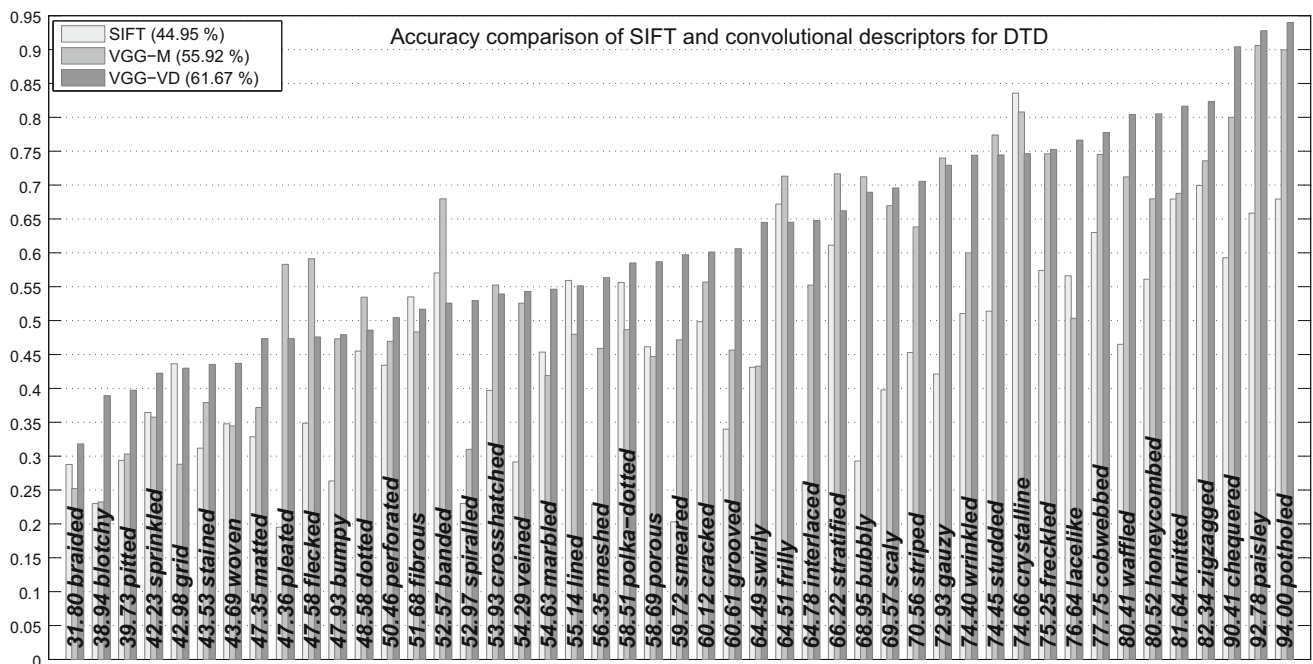

Fig. 5 Per class classification accuracy in the DTD data comparing three local image descriptors: SIFT, VGG-M, and VGG-VD. For all three local descriptors, BoVW with 4096 visual words was used. Classes are sorted by increasing BoVW-CNN-VD accuracy (this number is reported along each *bar*)

much smaller codebook as these representations multiply the dimensionality of the descriptors (Sect. 6.1.1). Since SIFT and CNN features are respectively 128 and 512-dimensional, K is set to 256 and 64 respectively. The impact of varying the number of visual words in the FV representation is further analyzed in Sect. 6.1.5.

Before pooling local descriptors with a FV, these are usually de-correlated by using PCA whitening. Here PCA is applied to SIFT, additionally reducing its dimension to 80, as this was empirically shown to improve recognition performance. The effect of PCA-reduction to the convolutional

features is studied in Sect. 6.1.7. The improved version of the FV is used in all the experiments (Sect. 3), and, similarly, for VLAD, we applied signed square root to the resulting encoding, which is then normalized component-wise (Sect. 4.2.3).

6.1.4.2 Analysis Results are reported in Table 3. In term of orderless encoders, BoVW and LLC result in similar performance for SIFT, while the difference is slightly larger and in favor of LLC for CNN features. Note that BoVW is used with the Hellinger kernel, which contributes to reducing the gap between BoVW and LLC. IFV and VLAD significantly

Table 3 Pooling encoder comparisons

Dataset	Meas. (%)	SIFT				VGG-M				VGG-M
		BoVW	LLC	VLAD	IFV	BoVW	LLC	VLAD	IFV	FC
DTD	acc	49.0 ± 0.8	48.2 ± 1.4	54.3 ± 0.8	58.6 ± 1.2	61.2 ± 1.3	64.0 ± 1.3	**67.6 ± 0.7**	66.8 ± 1.5	58.7 ± 0.9
OS+R	acc	30.0	30.8	32.5	39.8	41.3	45.3	49.7	**52.5**	41.3
KTH-T2b	acc	57.6 ± 1.5	56.8 ± 2.0	64.3 ± 1.3	70.2 ± 1.6	73.6 ± 2.8	**74.0 ± 3.3**	72.2 ± 4.7	73.3 ± 4.8	71.0 ± 2.3
FMD	acc	50.5 ± 1.7	48.4 ± 2.2	54.0 ± 1.3	59.7 ± 1.6	67.9 ± 2.2	71.7 ± 2.1	**74.2 ± 2.0**	73.5 ± 2.0	70.3 ± 1.8
VOC07	mAP11	51.2	47.8	56.9	59.9	72.9	75.5	76.8	76.4	**76.8**
MIT Indoor	acc	47.7	39.2	51.0	54.9	69.1	68.9	71.2	**74.2**	62.5

The table compares the orderless pooling encoders BoVW, LLC, VLAD, and IFV with either SIFT local descriptors and VGG-M CNN local descriptors (FV-CNN). It also compares pooling convolutional features with the CNN fully connected layers (FC-CNN). The table reports classification accuracies for all datasets except VOC07 and OS+R for which mAP-11 (Everingham et al. 2007) and mAP are reported, respectively

Bold values indicate the best results achieved by the compared methods

outperform BoVW and LLC in almost all tasks; FV is definitely better than VALD with SIFT features and about the same with CNN features. CNN features maintain a healthy lead on SIFT features regardless of the encoder used. Importantly, VLAD and FV (and to some extent BoVW and LLC) perform either substantially better or as well as the original FC encoders. Some of these observations can are confirmed by other experiments such as Table 4.

Next, we compare using CNN features with an orderless encoder (FV-CNN) as opposed to the standard FC layer (FC-CNN). As seen in Tables 3 and 4, in PASCAL VOC and MIT Indoor the FC-CNN descriptor performs very well but in line with previous results for this class of methods (Chatfield et al. 2014). FV-CNN performs similarly to FC-CNN in PASCAL VOC, KTH-T2b and FMD, but substantially better for DTD, OS+R, and MIT Indoor (e.g. for the latter +5 % for VGG-M and +13 % for VGG-VD).

As a sanity check, results are within 1 % of the ones reported in Chatfield et al. (2011) and Chatfield et al. (2014) for matching experiments on FV-SIFT and FC-VGG-M. The differences in case of SIFT LLC and BoVW are easily explained by the fact that, differently from Chatfield et al. (2011), our present experiments do not use SPP and image augmentation.

6.1.4.3 Conclusions The conclusions of these experiments are that: (i) IFV and VLAD are preferable to other orderless pooling encoders, that (ii) orderless pooling encoders such as the FV are at least as good and often significantly better than FC pooling with CNN features.

6.1.5 CNN Descriptor Variants Comparison

This section conducts additional experiments on CNN local descriptors to find the best variants.

6.1.5.1 Experimental Setup The same setup of the previous section is used. We compare the performance of FC-CNN and FV-CNN local descriptors obtained from VGG-M, VGG-VD as well as the simpler AlexNet (Krizhevsky et al. 2012) CNN which is widely adopted in the literature.

6.1.5.2 Analysis Results are presented in detail in Table 4. Within that table, the analysis here focuses mainly on texture and material datasets, but conclusions are similar for the other datasets. In general, VGG-M is better than AlexNet and VGG-VD is substantially better than VGG-M (e.g. on FMD, FC-AlexNet obtains 64.8 %, FC-VGG-M obtains 70.3 % (+5.5 %), FC-VGG-VD obtains 77.4 % (+7.1 %)). However, switching from FC to FV pooling improves the performance more than switching to a better CNN (e.g. on DTD going from FC-VGG-M to FC-VGG-VD yields a 7.1 % improvement, while going from FC-VGG-M to FV-VGG-M

Table 4 State of the art texture descriptors

dataset	Meas. (%)	SIFT FV	AlexNet FC	AlexNet FV	AlexNet FC+FV	VGG-M FC	VGG-M FV	VGG-M FC+FV	VGG-VD FC	VGG-VD FV	VGG-VD FC+FV	FV-SIFT FC+FV−VD	SoA
(a)													
CUReT	acc	99.0 ± 0.2	94.4 ± 0.4	98.5 ± 0.2	99.0 ± 0.2	94.2 ± 0.3	98.7 ± 0.2	99.1 ± 0.2	94.5 ± 0.4	99.0 ± 0.2	99.2 ± 0.2	**99.7 ± 0.1**	99.8 ± 0.1 Sifre and Mallat (2013)
UMD	acc	99.1 ± 0.5	95.9 ± 0.9	**99.7 ± 0.2**	99.7 ± 0.3	97.2 ± 0.9	**99.9 ± 0.1**	**99.8 ± 0.2**	97.7 ± 0.7	**99.9 ± 0.1**	**99.9 ± 0.1**	**99.9 ± 0.1**	99.7 ± 0.3 Sifre and Mallat (2013)
UIUC	acc	96.6 ± 0.8	91.1 ± 1.7	99.2 ± 0.4	99.3 ± 0.4	94.5 ± 1.4	**99.6 ± 0.4**	99.6 ± 0.3	97.0 ± 0.7	**99.9 ± 0.1**	**99.9 ± 0.1**	**99.9 ± 0.1**	99.4 ± 0.4 Sifre and Mallat (2013)
KT	acc	**99.5 ± 0.5**	95.5 ± 1.3	**99.6 ± 0.4**	**99.8 ± 0.2**	96.1 ± 0.9	**99.8 ± 0.2**	**99.9 ± 0.1**	97.9 ± 0.9	99.8 ± 0.2	**99.9 ± 0.1**	**100**	99.4 ± 0.4 Sifre and Mallat (2013)
ALOT	acc	94.6 ± 0.3	86.0 ± 0.4	**96.7 ± 0.3**	**97.8 ± 0.2**	88.7 ± 0.5	**97.8 ± 0.2**	**98.4 ± 0.1**	90.6 ± 0.4	**98.5 ± 0.1**	99.0 ± 0.1	**99.3 ± 0.1**	95.9 ± 0.5 Sulc and Matas (2014)
(b)													
KTH-T2b	acc	70.8 ± 2.7	71.5 ± 1.3	69.7 ± 3.2	72.1 ± 2.8	71 ± 2.3	73.3 ± 4.7	73.9 ± 4.9	75.4 ± 1.5	**81.8 ± 2.5**	81.1 ± 2.5	**81.5 ± 2.0**	76.0 ± 2.9 Sulc and Matas (2014)
FMD	acc	59.8 ± 1.6	64.8 ± 1.8	67.7 ± 1.5	71.4 ± 1.7	70.3 ± 1.8	73.5 ± 2.0	76.6 ± 1.9	77.4 ± 1.8	79.8 ± 1.8	**82.4 ± 1.5**	**82.2 ± 1.4**	57.7 ± 1.7 Sharan et al. (2013)
OS+R	acc	39.8	36.8	46.1	49.8	41.3	52.5	54.9	43.4	59.5	60.9	58.7	–
(c)													
DTD	acc	58.6 ± 1.2	55.1 ± 0.6	62.9 ± 1.4	66.5 ± 1.1	58.8 ± 0.8	**66.8 ± 1.6**	**69.8 ± 1.1**	62.9 ± 0.8	**72.3 ± 1.0**	74.7 ± 1.0	**75.5 ± 0.8**	–
DTD	mAP	61.3 ± 1.1	57.7 ± 0.9	66.5 ± 1.4	70.5 ± 1.2	62.1 ± 0.9	70.8 ± 1.2	**74.2 ± 1.1**	67.0 ± 1.1	**76.7 ± 0.8**	79.1 ± 0.8	**80.4 ± 0.9**	–
DTD-J	mAP	59.6 ± 0.6	58.4 ± 0.7	65.0 ± 0.9	68.3 ± 0.9	62.8 ± 0.7	**69.8 ± 0.9**	**72.9 ± 0.9**	67.3 ± 0.9	**75.8 ± 0.6**	77.5 ± 0.8	**78.9 ± 0.7**	–
OSA+R	mAP	56.5	53.9	62.1	64.6	54.3	65.2	67.9	49.7	67.2	67.9	68.2	–
(d)													
MSRC+R	acc	85.7	83.6	91.7	94.9	85.0	95.4	96.9	79.4	97.7	98.8	99.1	–
MSRC+R	msrc-acc	92.0	84.1	95.0	97.3	84.0	97.6	98.1	82.0	99.2	99.6	99.5	–
VOC07	mAP11	59.9	74.0	73.1	76.8	76.8	76.4	79.5	81.7	84.9	85.1	84.5	85.2 Wei et al. (2014)
VOC07	mAP	60.2	76.0	75.0	79.0	79.2	78.7	82.3	84.6	88.6	88.5	87.9	85.2 Wei et al. (2014)
MIT Ind.	acc	54.9	58.6	69.7	71.6	62.5	74.2	74.4	67.6	**81.0**	80.3	80.0	70.8 Zhou et al. (2014)
CUB	acc	17.5	45.8	49	54.1	46.1	49.9	54.9	54.6	66.7	**67.3**	**65.4**	73.9* Zhang et al. (2014)
CUB+R	acc	27.7	54.5	62.6	65.2	56.5	65.5	68.1	62.8	73.0	74.9	73.6	76.37 Zhang et al. (2014)

The table compares FC-CNN, FV-CNN on three networks trained on ImageNet— VGG-M, VGG-VD and AlexNet, and IFV on dense SIFT. We evaluated these descriptors on *a* texture datasets—in controlled settings, *b* material datasets (FMD, KTH-T2b, OS+R), *c* texture attributes (DTD, OSA+R) and *d* general categorisation datasets (MSRC+R, VOC07, MIT Indoor) and fine grained categorisation (CUB, CUB+R). For this experiment the region support is assumed to be known (and equal to the entire image for all the datasets except OS+R and MSRC+R and for CUB+R, where it is set to the bounding box of a bird)

* using a model without parts like ours the performance is 62.8 %

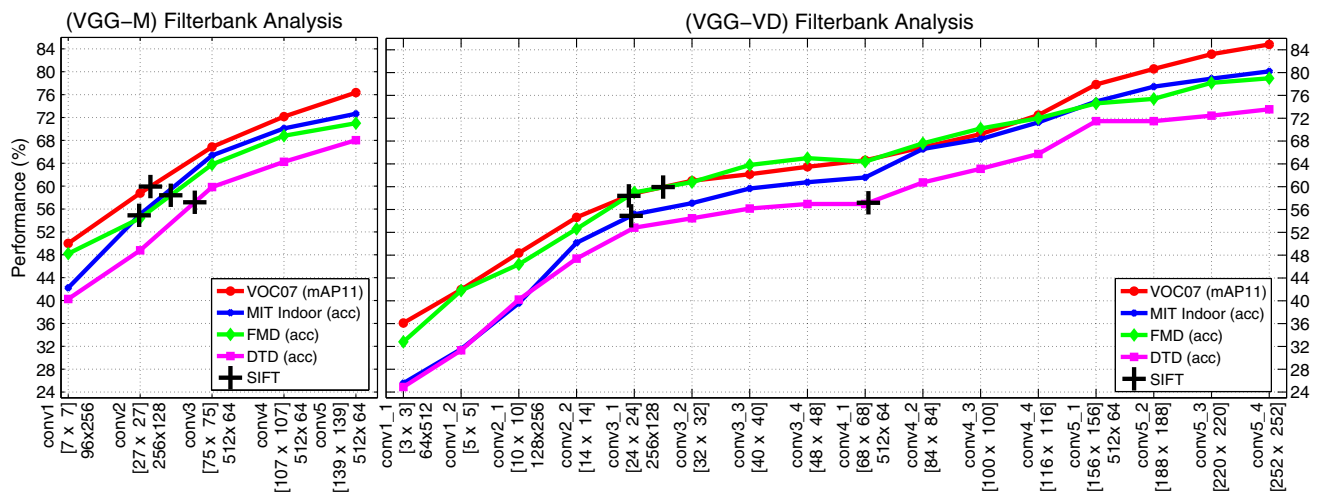Bold values indicate the results outperforming the existing state-of-the-art

Fig. 6 Effect of the depth on CNN features. The figure reports the performance of VGG-M (*left*) and VGG-VD (*right*) local image descriptors pooled with the FV encoder. For each layer the figures shows the size of the receptive field of the local descriptors (denoted $[N \times N]]$), as well as, for some of the layers, the dimension D of the local descriptors and the number K of visual words in the FV representation (denoted as $D \times K$). Curves for PASCAL VOC, MIT Indoor, FMD, and DTD are reported; the performance of using SIFT as local descriptors is reported as a *plus* (+) mark

yields a 11.3 % improvement). Combining FV-CNN and FC-CNN (by stacking the corresponding image representations) improves the accuracy by 1–2 % for VGG-VD, and up to 3–5 % for VGG-M. There is no significant benefit from adding FV-SIFT as well, as the improvement is at most 1 %, and in some cases (MIT, FMD) it degrades the performance.

Next, we analyze in detail the effect of depth on the convolutional features. Figure 6 reports the accuracy of VGG-M and VGG-VD on several datasets for features extracted at increasing depths. The pooling method is fixed to FV and the number of Gaussian centers K is set such that the overall dimensionality of the descriptor $2K D_k$ is constant. For both VGG-M and VGG-VD, the improvement with increasing depth is substantial and the best performance is obtained by the deepest features (up to 32 % absolute accuracy improvement in VGG-M and up to 48 % in VGG-VD). Performance increases at a faster rate up to the third convolutional layer (conv3) and then the rate tapers off somewhat. The performance of the earlier levels in VGG-VD is much worse than the corresponding layers in VGG-M. In fact, the performance of VGG-VD matches the performance of the deepest (fifth) layer in VGG-M in correspondence of conv5_1, which has depth 13.

Finally, we look at the effect of the number of Gaussian components (visual words) in the FV-CNN representation, testing possible values in the range 1 to 128 in small (1-16) increments. Results are presented in Fig. 7. While there is a substantial improvement in moving from one Gaussian component to about 64 (up to +15 % on DTD and up to 6 % on OS), there is little if any advantage at increasing the number of components further.
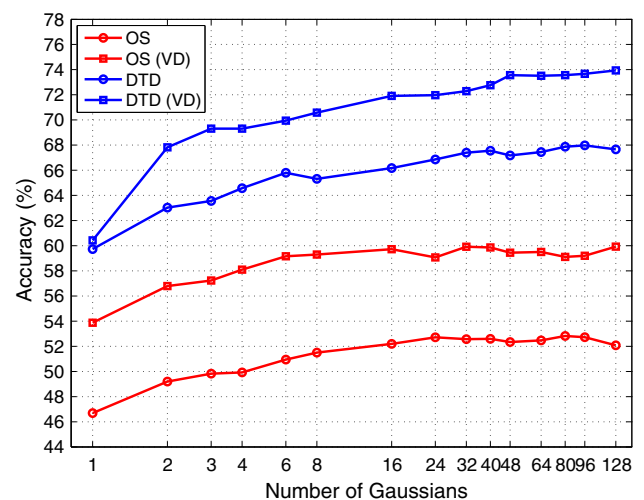

Fig. 7 Effect of the number of Gaussian components in the FV encoder. The figure shows the performance of the FV-VGG-M and FV-VGG-VD representations on the OS and DTD datasets when the number of Gaussians components in the GMM is varied from 1 to 128. Note that the abscissa is scaled logarithmically

6.1.5.3 Conclusions The conclusions of these experiments are as follows: (i) deeper models substantially improve performance; (ii) switching from FC to FV pooling has an ever more substantial impact, particularly for deeper models; (iii) combining FC and FV pooling has a modest benefit and there is no benefit in integrating SIFT features; (iv) in very deep models, most of the performance gain is realized in the very last few layers.

Table 5 The table the single and multi-scale variants of FC-CNN and FV-CNN using two deep CNN, VGG-M and VGG-VD, trained on the ImageNet ILSVRC, and a number of representative target datasets

dataset	Meas.	VGG-M				VGG-VD			
	(%)	FC (SS)	FC (MS)	FV (SS)	FV (MS)	FC (SS)	FC (MS)	FV (SS)	FV (MS)
KTH-T2b	acc	71 ± 2.3	68.9 ± 3.9	69.0 ± 2.8	73.3 ± 4.7	75.4 ± 1.5	75.1 ± 3.8	74.5 ± 4.4	81.8 ± 2.5
FMD	acc	70.3 ± 1.8	69.3 ± 1.8	71.6 ± 2.4	73.5 ± 2.0	77.4 ± 1.8	78.1 ± 1.7	79.4 ± 2.5	79.8 ± 1.8
DTD	acc	58.8 ± 0.8	59.9 ± 1.1	62.8 ± 1.5	66.8 ± 1.6	62.9 ± 0.8	65.3 ± 1.5	69.2 ± 0.8	72.3 ± 1.0
VOC07	mAP11	76.8	78	74.8	76.4	81.7	83.2	84.7	84.9
MIT Ind.	acc	62.5	66.1	68.1	74.2	67.6	75.3	76.8	81.0

The single scale variants are denoted FC (SS) and FV (SS) and the multi-scale variants as FC (MS) and FV (MS)

6.1.6 FV Pooling Versus FC Pooling

In the previous section, we have seen that switching from FC to FV pooling may have a substantial impact in certain problems. We could find three reasons that can explain this difference.

The first reason is that orderless pooling in FV can be more suitable for **texture modeling** than the order-sensitive pooling in FC. However, this explains the advantage of FV in texture recognition but not in object recognition.

The second reason is that FV pooling may reduce **overfitting in domain transfer**. Pre-trained FC layers could be too specialized for the source domain (e.g. ImageNet ILSVRC) and there may not be enough training data in the target domain to retrain them properly. On the contrary, a linear classifier built on FV pooling is less prone to overfitting as it encodes a simpler, smoother classification function than a sequence of FC layers in a CNN. This is further investigated in Sect. 6.2.3.

The third reason is the ability to easily incorporate information from **multiple image scales**.

In order to investigate this hypothesis, we evaluated FV-CNN by pooling CNN descriptors at a single scale instead of multiple ones, for both VGG-M and VGG-VD models. For datasets like FMD, DTD and MIT Indoor, FV-CNN at a single scale still generally outperforms FC-CNN (columns FC (SS) and FV (SS) in Table 5), by up to 5.6 % for VGG-M, and by up to 9.1 % for VGG-VD; however, the difference is less marked as using a single scale in FV-CNN looses up to 3.8 % accuracy points and and in some cases the representations is overtaken by FC-CNN.

The complementary experiment, namely using multiple scales in FC pooling, is less obvious as, by construction, FC-CNN resizes the input image to a fixed resolution. However, we can relax this restriction by computing multiple FC representations in a sliding-window manner (also know as a "fully-convolutional" network). Then individual representations computed at multiple locations and, after resizing the image, at multiple scales can be averaged in a single representation vector. We refer to this as multi-scale FC pooling.

Multi-scale FC codes perform slightly better than single-scale FC in most (but not all) cases; however, the benefit of using multiple scales is not as large as for multi-scale FV pooling, which is still significantly better than multi-scale FC.

6.1.7 Dimensionality Reduction of the CNN Descriptors

This section explores the effect of applying dimensionality reduction to the CNN local descriptors before FV pooling.

This experiment investigates the effect of two parameters, the number of Gaussians in the mixture model used by the FV encoder, and the dimensionality of the convolutional features, which we reduce using PCA. Various local descriptor dimensions are evaluated, from 512 (no PCA) to 32, reporting mAP on PASCAL VOC 2007, as a function of the pooled descriptor dimension. The latter is equal to $2KD$, where K is the number of Gaussian centers, and D the dimensionality of the local descriptor after PCA reduction.

Results are presented in Fig. 8 for VGG-M and VGG-VD. It can be noted that, for similar values of the total representation dimensionality $2KD$, the performance of PCA-reduced descriptors is a little better than not using PCA, provided that this is compensated by a large number of GMM components. In particular, similar to what was observed for SIFT in Perronnin et al. (2010), using PCA does improve the performance by 1–2 % mAP point; furthermore, reducing descriptors to 64 or 80 dimensions appears to result in the best performance.

6.1.8 Visualization of Descriptors

In this experiment we are interested in understanding which GMM components in the FV-CNN representation code for a particular concept, as well as in determining which areas of the input image contribute the most to the classification score.

In order to do so, let $\mathbf{w}$ be the weight vector learned by a SVM classifier for a target class using the FB-CNN representation as input. We partition $\mathbf{w}$ in subvectors $\mathbf{w}_k$, one for each
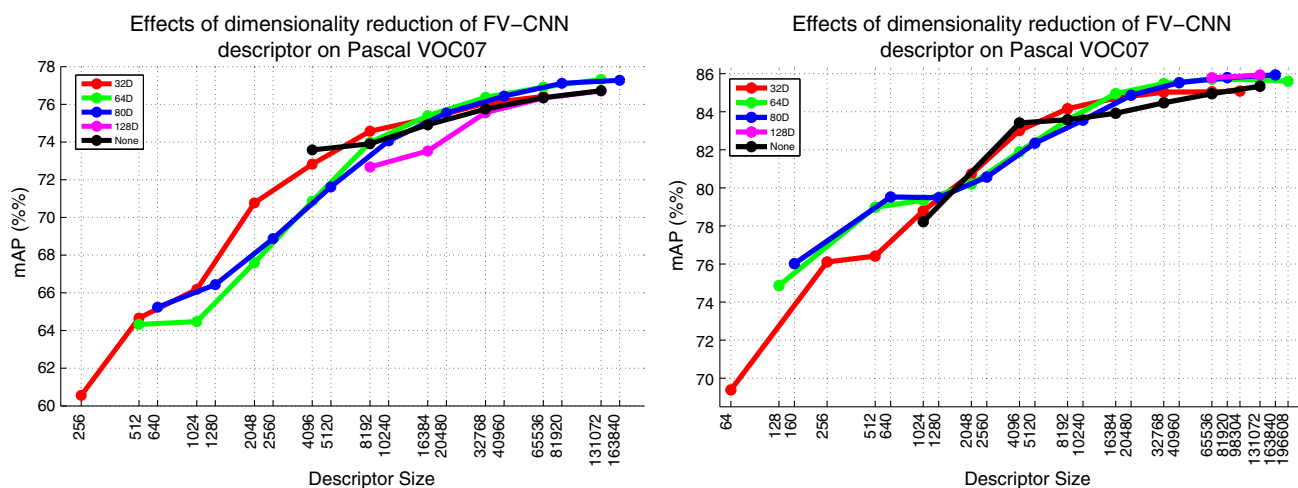
Fig. 8 PCA reduced FV-CNN. The figure reports the performance of VGG-M (*left*) and VGG-VD (*right*) local descriptors, on PASCAL VOC 2007, when reducing their dimensionality from 512 to up to 32 using PCA in combination with a variable number of GMM components. The *horizontal axis* report the total descriptor dimensionality, proportional to the dimensionality of the local descriptors by the number of GMM components

GMM component k, and rank components by decreasing value $\|\mathbf{w}_k\|$, matching the intuition that the GMM component that is predictive of the target class will result in larger weights. Having identified the top components for a target concept, the CNN local descriptors are then extracted from a test image, the descriptors that are assigned to a top component are selected, and their location is marked on the image. To simplify the visualization, features are extracted at a single scale.

As can be noted in Fig. 9 for some indicative texture types in DTD, the strongest GMM components do tend to fire in correspondence to the characteristic features of each texture. Hence, we conclude that GMM components, while trained in an unsupervised manner, contain clusters that can consistently localize features that capture distinctive characteristics of different texture types.

6.2 Evaluating Texture Representations on Different Domains

The previous section established optimal combinations of local image descriptors and pooling encoders in texture representations. This section investigates the applicability of these representations to a variety of domains, from texture (Sect. 6.2.1) to object and scene recognition (Sect. 6.2.3). It also emphasizes several practical advantages of orderless pooling compared to fully-connected pooling, including helping with the problem of domain shift in learned descriptors. This section focuses on problems where the goal is to either classify an image as a whole or a *known* region of an image, while texture segmentation is looked at later in Sect. 7.3.

6.2.1 Texture Recognition

Experiments on textures are divided in recognition in controlled conditions (Sect. 6.2.1.3), where the main sources of variability are viewpoint and illumination, recognition in the wild (Sect. 6.2.1.4), characterized by larger intra-class variations, and recognition in the wild and clutter (Sect. 6.2.1.5), where textures are a small portion of a larger scene.

6.2.1.1 Datasets and Evaluation Measures In addition to the datasets evaluated in Sect. 6.1, DTD, OS+R, FMD and KTH-T2b, we consider here also the standard benchmarks for texture recognition. CUReT (Dana et al. 1999) (5612 images, 61 classes), UIUC (Lazebnik et al. 2005) (1000 images, 25 classes), KTH-TIPS (Burghouts and Geusebroek 2009) (810 images, 10 classes) are collected in controlled conditions, by photographing the same instance of a material, under varying scale, viewing angle and illumination. UMD (Xu et al. 2009) consists of 1000 images, spread across 25 classes, but was collected in uncontrolled conditions. For these datasets, we follow the standard evaluation procedures, that is, we are using half of the images for training, and the remaining half for testing, and we are reporting accuracy, averaged over 10 random splits. The ALOT dataset (Burghouts and Geusebroek 2009) is similar to the existing texture datasets, but significantly larger, having 250 categories. For our experiments we used the protocol of Sulc and Matas (2014), using 20 images per class for training and the rest for testing.

6.2.1.2 Experimental Setup For the recognition tasks described in the following subsections, we compare SIFT, VGG-M, and VGG-VD local descriptors and the FC and FV
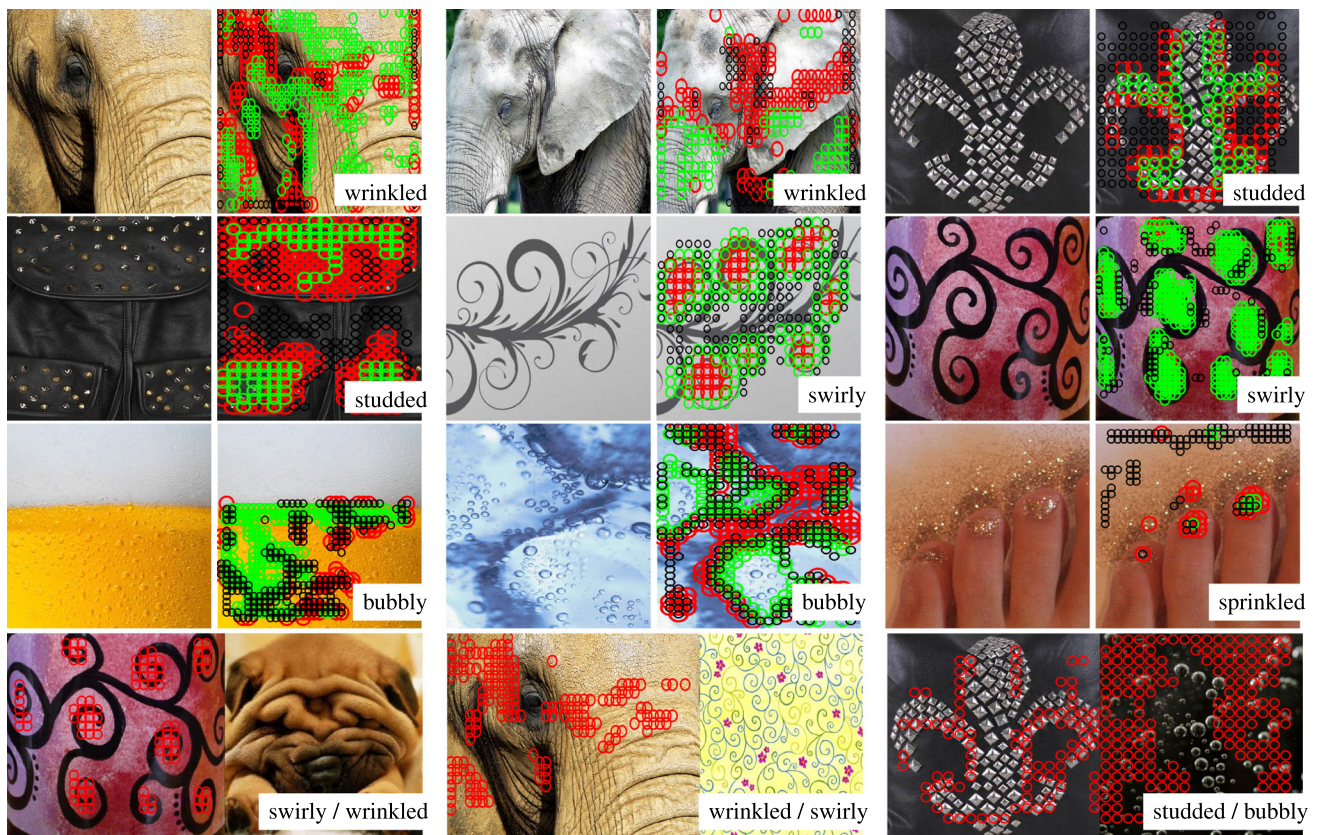
Fig. 9 FV-CNN descriptor visualization. *First three rows* each image shows the location of the CNN local descriptors that map to the FV-CNN components most strongly associated with the "wrinkled", "studded", "swirly", "bubbly", and "sprinkled" classes for a number of example images in DTD. *Red*, *green* and *black* marks correspond to the top three components selected as described in the text. *Last row* each image was obtained by combining two images, e.g. swirly and wrinkled, and we marked the CNN local descriptors associated with the first class. Swirly descriptors do not fire on the selected wrinkled images. The last pair, studded and bubbly is a harder, as the two images are visually similar, and the descriptors corresponding to *studded* appear on the bubbly image as well. In order to improve visibility, in these images, we show only the most discriminative FV component (Color figure online)

pooling encoders as these were determined before to be some of the best representative texture descriptors. Combinations of such descriptors are evaluated as well.

6.2.1.3 Texture Recognition in Controlled Conditions This paragraph evaluates texture representations on datasets which are collected under controlled condition (Table 4, Sect. a).

For instance recognition, CUReT, UIMD, UIUC are saturated by modern techniques such as Sifre and Mallat (2013); Sharma et al. (2012) and Sulc and Matas (2014), with accuracies above $\geq$ 99 %. There is little difference between methods, and FV-SIFT, FV-CNN, and FC-CNN behave similarly. KT is also saturated, although FC-CNN looses about (3 %) accuracy compared to FV-CNN.

In material recognition, KTH-T2b and ALOT offer a somewhat more interesting challenge. First, there is a significant difference between FC-CNN and FV-CNN (3–6 % absolute difference in KTH-T2b and 8–10 % in ALOT), consistent across all CNN evaluated. Second, CNN descriptors

are significantly better than SIFT on KTH-T2b and ALOT with absolute accuracy gains of up to 11 %.

Compared to the state of the art, FV-SIFT is generally very competitive. In KTH-T2b, FV-SIFT outperforms all recent methods (Cimpoi et al. 2014) with the exception of Sulc and Matas (2014) which is based on a variant of LBP. The latter is very strong in ALOT too, but in this case FV-SIFT is virtually as good. In the case of KTH-T2b, (Sulc and Matas 2014) is better than most of the deep descriptors as well, but it is still significantly bested by FV-VGG-VD (+5.5 %). Nevertheless, this is an example in which a specialized texture descriptor can be competitive with deep features, although of course deep features apply unchanged to several other problems.

On ALOT, FV-CNN with VGG-VD is on par with the result obtained by Badri et al. (2014)—98.45 %—but their model was trained with 30 images per class instead of 20. The same paper reports even better results, but when training with 50 images per class or by integrating additional synthetic training data.

6.2.1.4 Texture Recognition in the Wild This paragraph evaluates the texture representations on two texture datasets collected "in the wild": FMD (materials) and DTD (describable attributes).

Texture recognition in the wild is more comparable, in term of the type of intra-class variations, to object recognition than to texture recognition in controlled conditions. Hence, one can expect larger gains in moving from texture-specific descriptors to general-purpose descriptors. This is confirmed by the results. SIFT is competitive with AlexNet and VGG-M features in FMD (within 3 % accuracy), but it is significantly worse in DTD (+4.3 % for FV-AlexNet and +8.2 % for FV-VGG-M). FV-CNN is a little better than FC-CNN ($\sim$3 %) on FMD and substantially better in DTD ($\sim$8 %). Different CNN architectures exhibit very different performance; moving from AlexNet to VGG-VD, the accuracy absolute improvement is more than 11 % across the board.

Compared to the state of the art, FV-SIFT is generally very competitive, outperforming the specialized texture descriptors developed by Qi et al. (2014) and Sharan et al. (2013) in FMD (and this without using ground-truth texture segmentations as used by Sharan et al. (2013)). Yet FV-VGG-VD is significantly better than all these descriptors (+24.7 %).

In term of complementarity of the features, the combination of FC-CNN and FV-CNN improves performance by about 3 % across the board, but including FV-SIFT (labelled FV-SIFT/FC+FV-VD in the table) as well does not seem to improve performance further. This is in contrast with the fact that SIFT was found to be fairly complementary to FC-CNN on a variant of AlexNet in Cimpoi et al. (2014).

6.2.1.5 Texture Recognition Inclutter This section evaluates texture representations on recognizing texture materials and describable attributes in clutter. Since there is no standard benchmark for this setting, we introduce here the first analysis of this kind using the the OS+R and OSA+R datasets of Sect. 3.1. Recall that the +R suffix indicates that, while textures are imaged in clutter, the classifier is given the ground-truth region segmentation; therefore, the goal of this experiment is to evaluate the effect of realistic viewing conditions on texture recognition, but the problem of segmenting the textures is evaluated later, in Sect. 7.3.

Results are reported in Table 4 in sections b and c. As before, performance improves with the depth of CNNs. For example, in material recognition (OS+R) accuracy starts at about 39.1 % for FV-SIFT, is about the same for FC-VGG-M (41.3 %) and a little better for FC-VGG-VD (43.4 %). However, the benefit of switching from FC encoding to FV encoding is now even more dramatic. For example, on OS+R FV-VGG-M has accuracy 52.5 % (+11.2 %) while FV-VGG-VD 59.5 % (+16.1 %). This clearly demonstrates the advantage of orderless pooling of CNN local descriptors on FC pooling when regions of different sizes and shapes

must be evaluated. There is also a significant computational advantage (evaluated further in Sect. 6.2.3) if, as it is typical, several regions must be classified: in that case, CNN features need not to be recomputed for each region. Results on OSA+R are entirely analogous.

6.2.2 Object and Scene Recognition

This section evaluates texture descriptors on tasks other than texture recognition, namely coarse and fine-grained object categorization, scene recognition, and semantic region recognition.

6.2.2.1 Datasets and Evaluation Measures In addition to the datasets seen before, here we experiment with fine grained recognition in the CUB (Wah et al. 2011) dataset. This dataset contains 11788 images, representing 200 species of birds. The images are split approximately into half for training and half for testing, according to the list that accompanies the dataset. Image representations are either applied to the whole image (denoted CUB) or on the region counting the target bird using ground-truth bounding boxes (CUB+R). Performance in CUB and CUB+R is reported as per-image classification accuracy. For this dataset, the local descriptors are again extracted at multiple scales, but now only for the smaller range {0.5, 0.75, 1} which was found to work better for this task.

Performance is also evaluated on the MSRC dataset, designed to benchmark semantic segmentation algorithms. The dataset contains 591 images, for which some pixels are labelled with one of the 23 classes. In order to be consistent with the results reported in the literature, performance is reported in term of per-pixel classification accuracy, similar to the measure used for the OS task as defined in Sect. 3.1. However, this measure is further modified such that it is *not* normalized per class:

$$\text{acc-msrc}(\hat{c}) = \frac{|\{\mathbf{p} : c(\mathbf{p}) = \hat{c}(\mathbf{p})\}|}{|\{\mathbf{p} : c(\mathbf{p}) \neq 0\}|}. \tag{3}$$

6.2.2.2 Analysis Results are reported in Table 4 section d. On PASCAL VOC, MIT Indoor, CUB, and CUB+R the relative performance of the different descriptors is similar to what has been observed above for textures. Compared to the state-of-the-art results in each dataset, FC-CNN and particularly the FV-CNN descriptors are very competitive. The best result obtained in PASCAL VOC is comparable to the current state-of-the-art set by the deep learning method of Wei et al. (2014) (85.2 vs 84.9 % mAP), but using a much more straightforward pipeline. In MIT Places the best performance is also substantially superior (+10 %) to the current state-of-the-art using deep convolutional networks learned on the MIT Place dataset (Zhou et al. 2014) (this is discussed further below).

In the CUB dataset, the best performance is short (∼6 %) of the state-of-the-art results of Zhang et al. (2014). However, Zhang et al. (2014) uses a category-specific part detector and corresponding part descriptor as well as a CNN fine-tuned on the CUB data; by contrast, FV-CNN and FC-CNN are used here as *global image descriptors* which, furthermore, *are the same for all the datasets considered*. Compared to the results of Zhang et al. (2014) without part-based descriptors (but still using a part-based object detector), the best of our global image descriptors perform substantially better (62.1 vs 67.3 %).

Results on MSRC+R for semantic segmentation are entirely analogous; it is worth noting that, although ground-truth segments are used in this experiment and hence this number is not comparable with other reported in the literature, the best model achieves an outstanding 99.1 % per-pixel classification rate in this dataset.

6.2.2.3 Conclusions The conclusion of this section is that FV-CNN, although inspired by texture representations, are superior to many alternative descriptors in object and scene recognition, including more elaborate constructions. Furthermore, FV-CNN is significantly superior to FC-CNN in this case as well.

6.2.3 Domain Transfer

This section investigates in more detail the problem of domain transfer in CNN-based features. So far, the same underlying CNN features, trained on the ImageNet's ILSVCR data, were used in all cases. To investigate the effect of the source domain on performance, this section consider, in addition to these networks, new ones trained on the PLACES dataset (Zhou et al. 2014) to recognize scenes on a dataset of about 2.5 million labeled images. Zhou et al. (2014) showed that, applied to the task of scene recognition in MIT Indoor, these features outperform similar ones trained on ILSVCR [denoted CAFFE (Jia 2013) below]—a fact explained by the similarity of domains. We repeat this experiment using FC- and FV-CNN descriptors on top of VGG-M, VGG-VD, PLACES, and CAFFE.

Results are shown in Table 6. The FC-CNN performance is in line with those reported in Zhou et al. (2014)—in scene recognition with FC-CNN the same CNN architecture performs better if trained on the Places dataset instead of the ImageNet data (58.6 vs 65.0 % accuracy[7]). Nevertheless, stronger CNN architectures such as VGG-M and VGG-VD

Table 6 Accuracy of various CNNs on the MIT indoor dataset

CNN	Accuracy (%)		
	FC-CNN	FV-CNN	FC+FV-CNN
PLACES	65.0	67.6	73.1
CAFFE	58.6	69.7	71.6
VGG-M	62.5	74.2	74.4
VGG-VD	67.6	81.0	80.3

PLACES and CAFFE are the same CNN architecture ("AlexNet") but trained on different datasets (PLACES and ImageNet resp)
The domain specific advantage of training on PLACES dissapears when the convolutional features are used with FV pooling. For all architectures FV CNN outperformns FC and better architectures lead to better overall performance

can approach and outperform PLACES even if trained on ImageNet data (65.0 vs 62.5/67.6 %).

However, when it comes to using the filter banks with FV-CNN, conclusions are very different. First, FV-CNN outperforms FC-CNN in all cases, with substantial gains up to ∼11–12 % in correspondence of a domain transfer from ImageNet to MIT Indoor. The gap between FC-CNN and FV-CNN is the highest for VGG-VD models (67.6 vs 81.0 %, nearly 14 % difference), a trend also exhibited by other datasets as seen in Table 4. Second, *the advantage of using domain-specific CNNs disappears*. In fact, the same CAFFE model that is 6.4 % worse than PLACES with FC-CNN, is actually 2.1 % *better* when used in FV-CNN. The conclusion is that FV-CNN appears to be immune, or at least substantially less sensitive, to domain shifts.

Our explanation of this phenomenon is that the convolutional features are substantially less committed to a specific dataset than the fully connected layers. Hence, by using those, FV-CNN tends to be a lot more general than FC-CNN. A second explanation is that PLACES CNN may learn filters that tend to capture the overall spatial structure of the image, whereas CNNs trained on ImageNet tend to focus on localized attributes which may work well with orderless pooling.

Finally, we compare FV-CNN to alternative CNN pooling techniques in the literature. The closest method is the one of Gong et al. (2014), which uses a similar underlying CNN to extract local image descriptors and VLAD instead of FV for pooling. Notably, however, FV-CNN results on MIT Indoor are markedly better than theirs for both VGG-M and VGG-VD (68.8 vs 74.2 %/ 81.0 % resp.) and marginally better (69.7 %—Tables 4 and 6) when the same CAFFE CNN is used. Also, when using VLAD instead of FV for pooling the convolutional layer descriptors, the performance of our method is still better (68.8 % vs 71.2 %), as seen in Table 3. The key difference is that FV-CNN pools convolutional features, whereas (Gong et al. 2014) pools fully connected descriptors extracted from square image patches. Thus, even

without spatial information as used by Gong et al. (2014), FV-CNN is not only substantially faster—8.5× speedup when using the same network and three scales, but at least as accurate.

7 Experiments on Semantic Segmentation

The previous sections considered the problem of recognizing given image regions. This section explores instead the problem of automatically recognizing as well as segmenting such regions in the image.

7.1 Experimental Setup

Inspired by Cimpoi et al. (2014) that successfully ported object description methods to texture descriptors, here we propose a segmentation technique building on ideas from object detection. An increasingly popular method for object detection, followed for example by FC-CNN (Girshick et al. 2014), is to first propose a number of candidate object regions using low-level image cues, and then verifying a shortlist of such regions using a powerful classifier. Applied to textures, this requires a low-level mechanism to generate textured region proposals, followed by a region classifier. A key advantage of this approach is that it allows applying object-(FC-CNN) and texture-like (FV-CNN) descriptors alike. After proposal classification, each pixel can be assigned more than one label; this is solved with a simple voting schemes, also inspired by object detections methods.

The paper explores two such region generation methods: the *crisp regions* of Isola et al. (2014) and the *Multi-scale Combinatorial Grouping* (MCG) of Arbeláez et al. (2014). In both cases, region proposals are generated using low-level image cues, such as color or texture consistency, as specified by the original methods. It would of course be possible to incorporate FC-CNN and FV-CNN among these energy terms to potentially strengthen the region generation mechanism itself. However, this contradicts partially the logic of the scheme, which breaks down the problem into cheaply generating tentative segmentations and then verifying them using a more powerful (and likely expensive) model. Furthermore, and more importantly, these cues focus on separating texture *instances*, as presented in each particular image, whereas FC-CNN and FV-CNN are meant to identify a texture class. It is reasonable to expect instance-specific cues (say the color of a painted wall) to be better for segmentation.

The crisp region method generates a single partition of the image; hence, individual pixels are labelled by transferring the label of the corresponding region, as determined by the learned predictor. By contrast, MCG generates many thousands overlapping region proposals in an image and requires a mechanism to resolve potentially ambiguous pixel labellings. This is done using the following simple scheme. For each proposed region, its label is set to the the highest scoring class based on the multi-class SVM, and its score to the corresponding class score divided by the region area. Proposals are then sorted by increasing score and "pasted" to the image sequentially. This has the effect of considering larger regions before smaller ones and more confident regions after less confident ones for regions of the same area.

7.2 Dense-CRF Post-processing

The segmentation results delivered by the previous methods can potentially be hampered by the occasional failures of the respective front-end superpixel segmentation modules. But we can see the front-end segmentation as providing as a convenient way of pooling discriminative information, which can then be refined post-hoc through a pixel-level segmentation algorithm.

In particular, a series of recent works Chen et al. (2014), Bell et al. (2014) and Zheng et al. (2015) have reported that substantial gains can be obtained by combining CNN classification scores with the densely-connected Conditional Random Field (Dense-CRF) of Krähenbühl and Koltun (2011). Apart from its ability to incorporate information pertaining to image boundaries and color similarity, the Dense-CRF is particularly efficient when used in conjunction with approximate probabilistic inference: the message passing updates under a fully decomposable mean field approximation can be expressed as convolutions with a Gaussian kernel in feature space, implemented efficiently using high-dimensional filtering (Adams et al. 2010).

Inspired by these advances, we have employed the Dense-CRF segmentation algorithm *post-hoc*, with the aim of enhancing our algorithm's ability to localize region boundaries by taking context and low-level image information into account. For this we turn the superpixel classification scores into pixel-level unary terms, interpeting the SVM classifier's scores as indicating the negative energy associated to labelling each pixel with the respective labels. Even though Platt scaling could be used to turn the SVM scores into log-probability estimates, we prefer to estimate the transformation by jointly cross-validating the SVM-Dense-CRF cascade's parameters. In particular, similarly to Krähenbühl and Koltun (2011) and Chen et al. (2014), we set the dense CRF hyperparameters by cross-validation, performing grid search to find the values that perform best on a validation set.

7.3 Analysis

Results are reported in Table 7. Two datasets are evaluated: OS for material recognition and MSRC for things & stuff. Compared to OS+R, classifying crisp regions results in a

Table 7 Segmentation and recognition using crisp region proposals of materials (OS) and things & stuff (MSRC)

Dataset	Measure (%)	VGG-M			VGG-VD				
		FC-CNN	FV-CNN	FV+FC-CNN	FC-CNN	FV-CNN	FC+FV-CNN	CRF	SoA
OS	pp-acc	36.0	48.6 (46.9)	49.8	38.5	55.5 (55.7)	55.9	56.5	–
OSA	acc-osa (2)	42.8	66.0	63.4	42.1	67.9	64.6	68.9	–
MSRC	acc-msrc (3)	56.1	82.3	75.2	57.7	86.9	81.5	90.2	86.5 Ladicky et al. (2010)

Per-pixel accuracies are reported, using the MSRC variant (see text) for the MSRC dataset. Results using MCG proposals (Arbeláez et al. 2014) are reported in brackets for FV-CNN
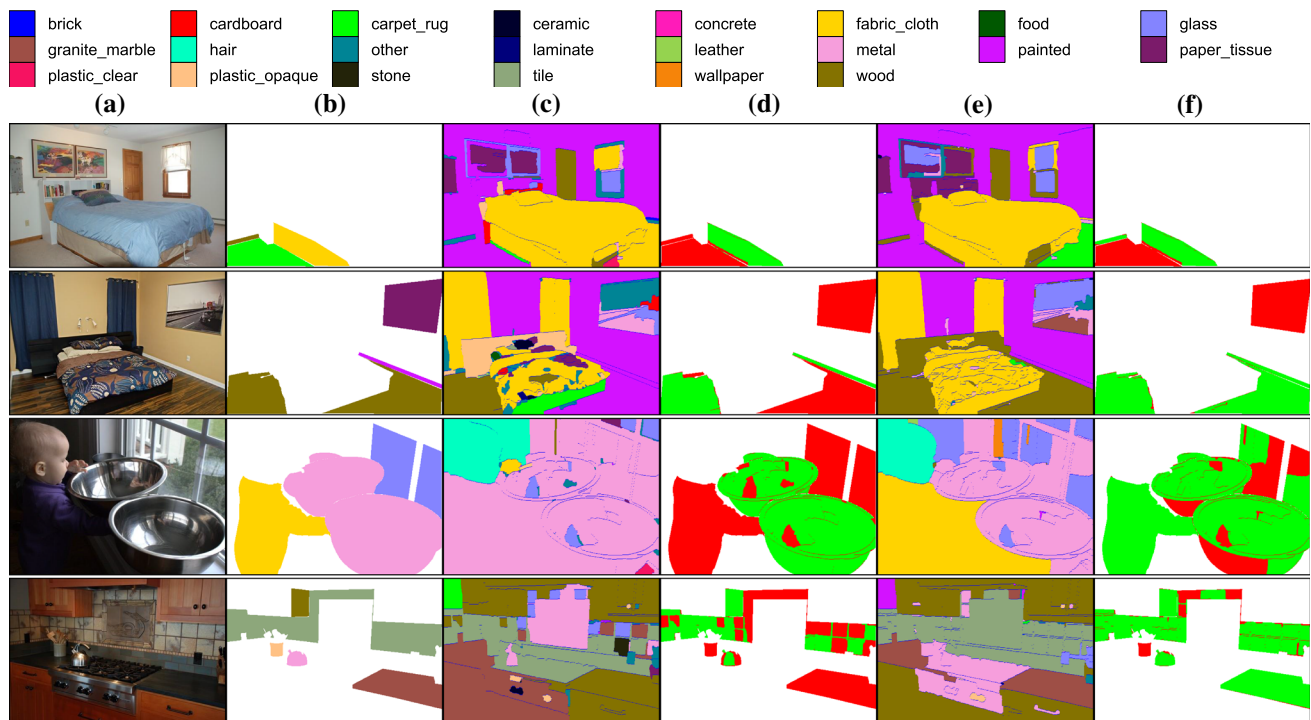
Fig. 10 OS material recognition results. Example test image with material recognition and segmentation on the OS dataset. **a** Original image. **b** Ground truth segmentations from the OpenSurfaces repository (note that not all pixels are annotated). **c** FC-CNN and crisp-region proposals segmentation results. **d** Correctly (*green*) and incorrectly (*red*) predicted pixels (restricted to the ones annotated). **e–f** the same, but for FV-CNN (Color figure online)

drop of about 10 % per-pixel classification accuracy for all descriptors.

At the same time, it shows that there is ample space for future improvements. In MSRC, the best accuracy is 87.0 %, just a hair above the best published result 86.5 % (Ladicky et al. 2010). Remarkably, these algorithms do not use any dataset-specific training, nor CRF-regularised semantic inference: they simply greedily classify regions as obtained from a general-purpose segmentation algorithms. CRF post-processing improves the results even further, up to 90.2 % in MSRC. Qualitative segmentation results (sampled at random) are given in Figs. 10 and 11.

Results using FV-CNN shown in Table 7 in brackets (due to the requirement of computing CNN features from scratch for every region, it was impractical to use FC-CNN with MCG proposals). The results are comparable to those using crisp regions, resulting in 55.7 % accuracy on the OS dataset. Other schemes such as non-maximum suppression of overlapping regions that are quite successful for object segmentation (Hariharan et al. 2014) performed rather poorly in this case. This is probably because, unlike objects, texture information is fairly localized and highly irregularly shaped in an image.

While for recognizing textures, materials or objects covering the entire image, the advantage in computational cost of FV-CNN on FC-CNN and was not significant, the latter consisting in evaluating few layers less, the advantage of
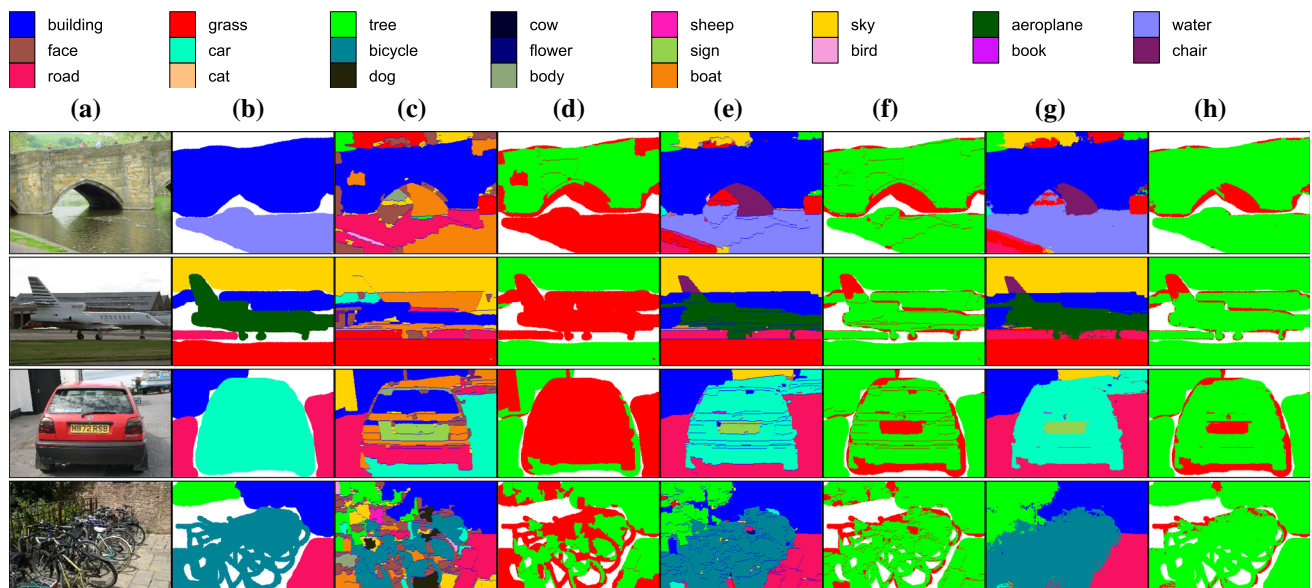
building	grass	tree	cow	sheep	sky	aeroplane	water
face	car	bicycle	flower	sign	bird	book	chair
road	cat	dog	body	boat			

(a)   **(b)**   **(c)**   **(d)**   **(e)**   **(f)**   **(g)**   **(h)**

Fig. 11 MSRC object segmentation results. **a** Image, **b** ground-truth, **c–d** FC-CNN segmentation and errors, **e–f** FV-CNN segmentation and errors (in *red*), **g–h** segmentation and errors after Dense CRF post-processing (Color figure online)

Table 8 DTD for material recognition

DTD classifier	KTH-T2b		FMD	
Method	Linear	RBF	Linear	RBF
FV-SIFT	64.74 ± 2.36	67.75 ± 2.89	49.24 ± 1.73	52.53 ± 1.26
FV-CNN	67.39 ± 3.75	67.66 ± 3.30	62.81 ± 1.33	64.69 ± 1.41
FV-CNN-VD	74.59 ± 2.45	74.71 ± 1.96	$\mathbf{70.81 \pm 1.39}$	$\mathbf{73.09 \pm 1.35}$
FV-SIFT + FC-CNN	73.98 ± 1.24	74.53 ± 1.14	64.20 ± 1.65	67.13 ± 1.95
FV-SIFT + FC-CNN-VD	$\mathbf{74.52 \pm 2.31}$	$\mathbf{77.14 \pm 1.36}$	69.21 ± 1.77	72.17 ± 1.66
Previous best	76.0 ± 2.9 Sulc and Matas (2014)		57.7 ± 1.7 Sharan et al. (2013) and Qi et al. (2014)	

Accuracy on material recognition on the KTH-T2b and FMD benchmarks obtained by using as image representation the predictions of the 47 DTD attributes by different methods: FV-SIFT, FV-CNN (using either VGG-M or VGG-VD) or combinations
Accuracies are compared to published state of the art results
Bold values indicate the best results achieved by the compared methods

FV-CNN becomes clear for segmentation tasks, as FC-CNN requires recomputing the features for every region proposal.

8 Applications of Describable Texture Attributes

This section explores two applications of the DTD attributes: using them as general-purpose texture descriptors (Sect. 8.1) and as a tool for search and visualization (Sect. 8.2).

8.1 Describable Attributes as Generic Texture Descriptors

This section explores using the 47 describable attributes of Sect. 2 as a general-purpose texture descriptor. The first step in this construction is to learn a multi-class predictor for the 47 attributes; this predictor is trained on DTD using a texture representation of choice and a multi-class linear SVM as before. The second step is to evaluate the multi-class predictor to obtain a 47-dimensional descriptor (of class scores) for each image in a target dataset. In this manner, one obtains a novel and very compact representation which is then used to learn a multi-class non-linear SVM classifier, for example for material recognition.

Results are reported in Table 8 for material recognition in FMD and KTH-T2b. There are two important factors in this experiment. The first one is the choice of the DTD attributes predictor. Here the best texture representations found before are evaluated: FV-SIFT, FC-CNN, and FV-CNN (using either VGG-M or VGG-VD local descriptors), as well as their combinations. The second one is the choice of classifier used to predict a texture material based on the 47-dimensional vector

of describable attributes. This is done using either a linear or RBF SVM.

Using a linear SVM and FV-SIFT to predict the DTD attributes yields promising results: 64.7 % classification accuracy on KTH-T2b and 49.2 % on FMD. The latter outperforms the specialized aLDA model of Sharan et al. (2013) combining color, SIFT and edge-slice features, whose accuracy is 44.6 %. Replacing SIFT with CNN image descriptors (FV-CNN) improves results significantly for FMD (49.2 vs 62.8 % for VGG-M and 70.8 % for VGG-VD) as well as KTH-T2b (64.7 vs 67.4 and 74.6 % respectively). While these results are not as good as using the best texture representations directly on these datasets, remarkably the dimensionality of the DTD descriptors is *two orders of magnitude smaller* than all the other alternatives (Sharan et al. 2013; Qi et al. 2014).

An advantage of the small dimensionality of the DTD descriptors is that using an RBF classifier instead of the linear one is relatively cheap. Doing so improves the performance by 1–3 % on both FMD and KTH-T2b across experiments. Overall, the best result of the DTD features on KTH-T2b is 77.1 % accuracy, slightly better than the state-of-the-art accuracy rate of 76.0 % of Sulc and Matas (2014). On FMD the DTD features outperform significantly the state of the art []: 72.17 % accuracy vs. 57.7 %, an improvement of about 15 %.

The final experiment compares the semantic attributes of Matthews et al. (2013) on the Outex data. Using FV-SIFT and a linear classifier to predict the DTD attributes, performance on the retrieval experiment of Matthews et al. (2013) is 49.82 % mAP which is not competitive with their result of 63.3 % obtained using LBP^u (Sect. 4.1). To verify whether this was due to LBP^u being particularly optimized for the

Outex data, the DTD attributes where trained again using FV on top of the LBP^u local image descriptors; by doing so, using the 47 attributes on Outex results in an accuracy of 64.5 % mAP; at the same time, Table 2 shows that LBP^u is not a competitive predictor on DTD itself. This confirms the advantage of the LBP^u on the Outex dataset.

8.2 Search and Visualization

This section includes a short qualitative evaluation of the DTD attributes. Perhaps their most appealing property is interpretability; to verify that semantics transfer in a reasonable way across domains, Fig. 12 shows an excellent semantic correlation between the ten categories in KTH-T2b and the attributes in DTD. For example, aluminum foil is found to be *wrinkled*, while bread is found be *bumpy*, *pitted*, *porous* and *flecked*.

As an additional application of our describable texture attributes we compute them on a large dataset of 10,000 wallpapers and bedding sets from houzz.com. The 47 attribute classifiers are learned as in Sect. 6 using the FV-SIFT representation and then applied to the 10,000 images to predict the strength of association of each attribute and image. Classifier scores are re-calibrated on the target data and converted to probabilities by rescaling the scores to have a maximum value of one on the whole dataset. Figure 13 shows some example attribute predictions, selecting for each of a number of attributes an image that has a score close to 1 (excluding images used for calibrating the scores), and then including additional top two attribute matches. The top two matches tend to be a very good description of each texture or pattern, while the third is a good match in about half of the cases.
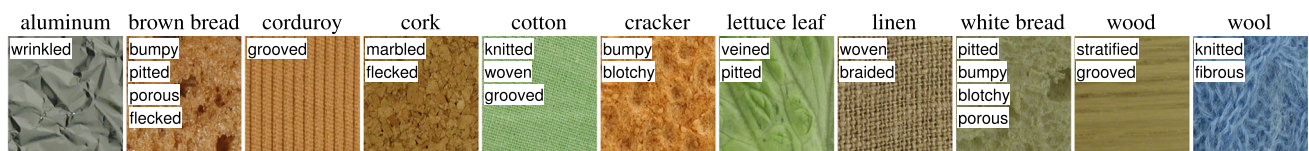

Fig. 12 Descriptions of materials from KTH-T2b dataset. These words are the most frequent top scoring texture attributes (from the list of 47 we proposed), when classifying the images from the KTH-T2b dataset. The descriptions are obtained by considering the whole material category, while a single image per material is shown for visualization

Fig. 13 Bedding sets (*top*) and wallpapers (*bottom*) with the top 3 attributes predicted by our classifier and normalized classification score in *brackets*

9 Conclusions

In this paper we have introduced a dataset of 5640 images collected "in the wild" that have been jointly labelled with 47 describable texture attributes and have used this dataset to study the problem of extracting semantic properties of textures and patterns, addressing real-world human-centric applications. We have also introduced a novel analysis of material and texture attribute recognition in a large dataset of textures in clutter derived from the excellent OpenSurfaces dataset. Finally, we have analyzed texture representation in relation to modern deep neural networks. The main finding is that orderless pooling of convolutional neural network features is a remarkably good texture descriptor, sufficiently versatile to dub as a scene and object descriptor too and resulting in the new state-of-the-art performance in several benchmarks.

References

Adams, A., Baek, J., & Davis, M. A. (2010). Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, *29*(2), 753–762.

Adelson, E. H. (2001). On seeing stuff: The perception of materials by humans and machines. *SPIE 4299*.

Amadasun, M., & King, R. (1989). Textural features corresponding to textural properties. *Systems, Man, and Cybernetics*, *19*(5), 1264–1274.

Arandjelovic, R., & Zisserman, A. (2012). Three things everyone should know to improve object retrieval. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Arbeláez, P., Pont-Tuset, J., Barron, J.T., Marques, F., & Malik, J. (2014). Multiscale combinatorial grouping. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Badri, H., Yahia, H., & Daoudi, K. (2014). Fast and accurate texture recognition with multilayer convolution and multifractal analysis. *Proceedings of ECCV*, *8689*, 505–519.

Bajcsy, R. (1973). Computer description of textured surfaces. In *Proceedings of the 3rd international joint conference on Artificial intelligence (IJCAI)*. Morgan Kaufmann Publishers Inc.

Bell, S., Upchurch, P., Snavely, N., & Bala, K. (2013). Opensurfaces: A richly annotated catalog of surface appearance. In *Proceedings of SIGGRAPH*.

Bell, S., Upchurch, P., Snavely, N., & Bala, K. (2014). Material recognition in the wild with the materials in context database. arXiv:1412.0623.

Bell, S., Upchurch, P., Snavely, N., & Bala, K. (2015). Material recognition in the wild with the materials in context database. In *Proceedings of computer vision and pattern recognition (CVPR)*

Berg, T., Berg, A., & Shih, J. (2010). Automatic attribute discovery and characterization from noisy web data. ECCV

Berlin, B., & Kay, P. (1991). Basic color terms: Their universality and evolution. Berkeley: University of California Press.

Bhushan, N., Rao, A., & Lohse, G. (1997). The texture lexicon: Understanding the categorization of visual texture terms and their relationship to texture images. *Cognitive Science*, *21*(2), 219–246.

Bourdev, L., Maji, S., & Malik, J. (2011). Describing people: A poselet-based approach to attribute classification. In *IEEE international conference on computer vision (ICCV)*.

Bovik, A. C., Clark, M., & Geisler, W. S. (1990). Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(1), 55–73.

Brodatz, P. (1966). *Textures: A photographic album for artists & designers*. New York: Dover.

Burghouts, G. J., & Geusebroek, J. M. (2009). Material-specific adaptation of color invariant features. *Pattern Recognition Letters*, *30*, 306–313.

Caputo, B., Hayman, E., & Mallikarjuna, P. (2005). Class-specific material categorisation. In *IEEE international conference on computer vision (ICCV)*.

Chatfield, K., Lempitsky, V., Vedaldi, A., & Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of BMVC*.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014) Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of BMVC*.

Chaudhuri, B., & Sarkar, N. (1995). Texture segmentation using fractal dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(1), 72–77.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A.L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. CoRR. arXiv:1412.7062.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., & Vedaldi, A. (2014) Describing textures in the wild. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Cimpoi, M., Maji, S., & Vedaldi, A. (2015). Deep filter banks for texture recognition and description. In: *Proceedings of computer vision and pattern recognition (CVPR)*.

Clarke, A. D. F., Halley, F., Newell, A. J., Griffin, L. D., & Chantler, M. J. (2011). Perceptual similarity: A texture challenge. In *Proceedings of BMVC*.

Csurka, G., Dance, C. R., Dan, L., Willamowski, J., & Bray, C. (2004) Visual categorization with bags of keypoints. In *Proceedings of ECCV workshop on statistics learning in computer vision*

Dana, K. J., van Ginneken, B., Nayar, S. K., & Koenderink, J. J. (1999). Reflectance and texture of real world surfaces. *ACM Transactions on Graphics*, *18*(1), 1–34.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009) ImageNet: A large-scale hierarchical image database. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Dunn, D., Higgins, W., & Wakeley, J. (1994). Texture segmentation using 2-d gabor elementary functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(2), 130–149.

Efros, A., & Leung, T. (1999). Texture synthesis by non-parametric sampling. In *Proceedings of computer vision and pattern recognition (CVPR)* (vol. 2)

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2008). The PASCAL visual object classes chal-

lenge 2008 (VOC2008) results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

Everingham, M., Zisserman, A., Williams, C., & Gool, L. V. (2007). *The PASCAL visual obiect classes challenge 2007 (VOC2007) results*. Pascal Challenge: Technical Report.

Farhadi, A., Endres, I., Hoiem, D., & Forsyth, D. (2009). Describing objects by their attributes. In *IEEE proceedings of computer vision and pattern recognition (CVPR)*. (pp. 1778–1785)

Ferrari, V., & Zisserman, A. (2007). Learning visual attributes. In *Proceedings of NIPS*.

Forsyth, D. (2001). Shape from texture and integrability. In *IEEE proceedings of ICCV*. (vol. 2, pp. 447–452)

Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *13*(9), 891–906.

Gårding, J. (1992). Shape from texture for smooth curved surfaces in perspective projection. *Journal of Mathematical Imaging and Vision*, *2*(4), 327–350.

Geusebroek, J. M., Smeulders, A. W. M., & van de Weijer, J. (2003). Fast anisotropic gauss filtering. *IEEE Transactions on Image Processing*, *12*(8), 938–943.

Girshick, R.B., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Gong, Y., Wang, L., Guo, R., & Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of ECCV*.

Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2014). Simultaneous detection and segmentation. In *Computer vision–ECCV 2014* (pp. 297–312). New York: Springer.

Hayman, E., Caputo, B., Fritz, M., & Eklundh, J.O. (2004). On the significance of real-world conditions for material classification. In *Proceedings of ECCV*.

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In: *Proceedings of ECCV*.

Isola, P., Zoran, D., Krishnan, D., & Adelson, E. H. (2014). Crisp boundary detection using pointwise mutual information. In: *Proceedings of ECCV*.

Jain, A., & Farrokhnia, F. (1991). Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, *24*(12), 1167–1186.

Jégou, H., Douze, M., Schmid, C., & Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Jia, Y. (2013) Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/.

Julesz, B. (1981). Textons, the elements of texture perception, and their interactions. *Nature*, *290*(5802), 91–97.

Julesz, B., & Bergen, J. R. (1983). Textons, the fundamental elements in preattentive vision and perception of textures. *Bell System Technical Journal*, *62*(6, Pt 3), 1619–1645.

Krähenbühl, P., & Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems 24: 25th annual conference on neural information processing systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain* (pp. 109–117)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*.

Kumar, N., Berg, A., Belhumeur, P., & Nayar, S. (2011). Describable visual attributes for face verification and image search. *Pattern Analysis & Machine Intelligence*, *33*(10), 1962–1977.

Ladicky, L., Russell, C., Kohli, P., & Torr, P. (2010). Graph cut based inference with co-occurrence statistics. In *Proceedings of ECCV* (pp. 239–253)

Lazebnik, S., Schmid, C., & Ponce, J. (2005). A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(8), 2169–2178.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *IEEE computer society conference on computer vision and pattern recognition*

Leung, T., & Malik, J. (1996) Detecting, localizing and grouping repeated scene elements from an image. In: *Proceedings of ECCV*. Lecture notes in computer science 1064 (pp. 546–555)

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

Leung, T., & Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, *43*(1), 29–44.

Liu, L., Wang, L., & Liu, X. (2011). In defense of soft-assignment coding. In *IEEE International Conference on Computer Vision (ICCV)* (pp. 2486–2493)

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In: *Proceedings of ICCV*.

Maji, S., Berg, A.C., & Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Malik, J., & Perona, P. (1990). Preattentive texture discrimination with early vision mechanisms. *JOSA A*, *7*(5), 923–932.

Malik, J., & Rosenholtz, R. (1997). Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, *23*(2), 149–168.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis & Machine Intelligence*, *11*, 674–693.

Mallikarjuna, P., Fritz, M., Targhi, A. T., Hayman, E., Caputo, B., & Eklundh, J.O. (2006). The KTH-TIPS and KTH-TIPS2 databases

Manjunath, B., & Chellappa, R. (1991). Unsupervised texture segmentation using markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(5), 478–482.

Matthews, T., Nixon, M. S., & Niranjan, M. (2013). Enriching texture analysis with semantic data. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, *29*(1), 51–59.

Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution grayscale and rotation invariant texture classification with local binary patterns. *Pattern Analysis & Machine Intelligence*, *24*(7), 971–987.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *Proceedings of computer vision and pattern recognition (CVPR)*

Oxholm, G., Bariya, P., & Nishino, K. (2012). The scale of geometric texture. In *European conference on computer vision* (pp. 58–71). Berlin: Springer

Parikh, D., & Grauman, K. (2011). Relative attributes. In *Proceedings of ICCV*.

Parkhi, O. M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). A compact and discriminative face track descriptor. In *Proceedings of computer vision and pattern recognition (CVPR)*

Patterson, G., & Hays, J. (2012). Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Proceedings of computer vision and pattern recognition (CVPR)*

Perronnin, F., & Dance, C. R. (2007). Fisher kernels on visual vocabularies for image categorization. In *Proceedings of computer vision and pattern recognition (CVPR)*

Perronnin, F., & Larlus, D. (2015). Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3743–3752)

Perronnin, F., Sánchez, J., & Mensink, T. (2010). Improving the Fisher kernel for large-scale image classification. In *Proceedings of ECCV*.

Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases.

Platt, J. C. (2000). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf, & D. Schuurmans (Eds.), *Advances in large margin classifiers*. Cambridge: MIT Press.

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

Qi, X., Xiao, R., Li, C. G., Qiao, Y., Guo, J., & Tang, X. (2014). Pairwise rotation invariant co-occurrence local binary pattern. *Pattern Analysis & Machine Intelligence*, 36(11), 2199–2213.

Quattoni, A., & Torralba, A. (2009). Recognizing indoor scenes. In *Proceedings of computer vision and pattern recognition (CVPR)*

Rao, A. R., & Lohse, G. L. (1996). Towards a texture naming system: Identifying relevant dimensions of texture. *Vision Research*, 36(11), 1649–1669.

Razavin, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. In *DeepVision workshop*.

Schwartz, G., & Nishino, K. (2013). Visual material traits: Recognizing per-pixel material context. In *Proceedings of CVCP*.

Sharan, L., Liu, C., Rosenholtz, R., & Adelson, E. H. (2013). Recognizing materials using perceptually inspired features. *International Journal of Computer Vision*, 103(3), 348–371.

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

Sharma, G., ul Hussain, S., & Jurie, F. (2012). Local higher-order statistics (lhs) for texture categorization and facial analysis. In *Proceedings of ECCV*.

Sifre, L., & Mallat, S. (2013). Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. CoRR. arXiv:1409.1556

Sivic, J., & Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. pp. 1470–1477.

Sulc, M., & Matas, J. (2014). Fast features invariant to rotation and scale of texture. Technical Report

Tamura, H., Mori, S., & Yamawaki, T. (1978). Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, 8(6), 460–473.

Timofte, R., & Van Gool, L. (2012). A training-free classification framework for textures, writers, and materials. In *Proceedings of BMVC*.

Varma, M., & Zisserman, A. (2003). Texture classification: Are filter banks necessary? In *IEEE proceedings of computer vision and pattern recognition (CVPR)* (vol. 2, pp. II–691)

Varma, M., & Zisserman, A. (2005). A statistical approach to texture classification from single images. *IJCV*, 62(1), 61–81.

Vedaldi, A., & Fulkerson, B. (2010). VLFeat—an open and portable library of computer vision algorithms.

Vedaldi, A., & Lenc, K. (2014). Matconvnet-convolutional neural networks for matlab. arXiv:1412.4564.

Vedaldi, A., & Zisserman, A. (2010). Efficient additive kernels via explicit feature maps. In *Proceedings of computer vision and pattern recognition (CVPR)*.

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wang, J., Markert, K., & Everingham, M. (2009). Learning models for object recognition from natural language descriptions. In *Proceedings of BMVC*.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010). Locality-constrained linear coding for image classification. In *IEEE Conference on computer vision and pattern recognition (CVPR)* (pp. 3360–3367)

Wang, L., & He, D. C. (1990). Texture classification using texture spectrum. *Pattern Recognition*, 23(8), 905–910.

Wei, L., & Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH*. (pp. 479–488). ACM Press/Addison-Wesley Publishing Co.

Wei, Y., Xia, W., Huang, J., Ni, B., Dong, J., Zhao, Y., & Yan, S. (2014). Cnn: Single-label to multi-label.

Xu, Y., Ji, H., & Fermuller, C. (2009). Viewpoint invariant texture description using fractal analysis. *IJCV*, 83(1), 85–100.

Zhang, N., Donahue, J., Girshickr, R., & Darrell, T. (2014). Part-based R-CNNs for fine-grained category detection. In *Proceedings of ECCV*

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., & Torr, P.H.S. (2015). Conditional random fields as recurrent neural networks. CoRR **abs/1502.03240**. arXiv:1502.03240.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Proceedings of NIPS*.

Zhou, X., Yu, K., Zhang, T., & Huang, T.S. (2010). Image classification using super-vector coding of local image descriptors. In *Computer Vision–ECCV* (pp. 141–154) New York: Springer.