Recognizing Describable Attributes and Materials of Textures in the Wild and Clutter

D.Phil Thesis

Robotics Research Group
Department of Engineering Science
University of Oxford

Supervisor:
Dr. Andrea Vedaldi

Mircea Cimpoi
St Edmund Hall

Hilary Term, 2015

Mircea Cimpoi
St Edmund Hall

Doctor of Philosophy
Hilary Term, 2015

Recognizing Describable Attributes and Materials of Textures in the Wild and Clutter

Abstract

Visual textures play an important role in image understanding because they are a key component of the semantic of many images. Furthermore, texture representations, which pool local image descriptors in an orderless manner, have had a tremendous impact in a wide range of computer vision problems, from texture recognition to object detection. In this thesis we make several contributions to the area of texture understanding.

First, we add a new semantic dimension to texture recognition. Instead of focusing on instance or material recognition, we propose a human-interpretable vocabulary of texture attributes, inspired from studies in Cognitive Science, to describe common texture patterns. We also develop a corresponding dataset, the *Describable Texture Dataset* (DTD), for benchmarking. We show that these texture attributes produce intuitive descriptions of textures. We also show that they can be used to extract a very low dimensional representation of any texture that is very effective in other texture analysis tasks, including improving the state-of-the art in material recognition on the most challenging datasets available today.

Second, we look at the problem of recognizing texture attributes and materials in realistic uncontrolled imaging conditions, including when textures appear in clutter. We build on top of the recently proposed Open Surfaces dataset, introduced by the graphics community, by deriving a corresponding benchmarks for material recognition. In addition to material labels, we also augment a subset of Open Surfaces with semantic attributes.

Third, we propose a novel texture representation, combining the recent advances in deep-learning with the power of Fisher Vector pooling. We provide thorough evaluation of the new representation, and revisit in general classic texture representations, including bag-of-visual-words, VLAD and the Fisher Vectors, in the context of deep learning. We show that these pooling mechanisms have excellent efficiency and generalisation properties if the convolutional layers of a deep model are used as local features. We obtain in this manner state-of-the-art performance in numerous datasets, both in texture recognition and image understanding in general. We show through our experiments that the proposed representation is an efficient way to apply deep features to image regions, and that it is an effective manner of transferring deep features from one domain to another.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Mircea Cimpoi, St Edmund Hall

Acknowledgements

First and foremost, I am the most grateful to my supervisor, Andrea Vedaldi, for the guidance, advice and constant encouraging, as well as the knowledge I was privileged to receive, and, of course, for the long and late hours spent in the whole supervision process.

I would also like to thank my collaborators and co-authors, Subhransu Maji and Iasonas Kokkinos, and to my examiners, for the constructive comments and feedback, which helped improving and making more clear the content of this thesis.

Probably I would not have applied to Oxford, without the advice of David Nister, to whom I am grateful for inspiring my decision, and motivating me to dream big and aim higher.

The lab colleagues and friends – past, present as well as visitors, with whom my path intersected – played an important role in making VGG the inspiring place it is.

Most importantly, I would like to thank my wife, for supporting, encouraging and especially understanding me in these three and a half years, and for doing her best to keep my motivation and enthusiasm levels high. And finally, I would like to thank my parents and family for understanding and support.

Contents

Chapter 1

Introduction

1.1 Objective

This thesis addresses the problem of *describing textures* in images with one or more *semantic attributes*. Texture patterns are a distinctive property of many natural and man-made objects; for example the wings of a butterfly can be dotted, the skin of a snake scaly, and a shirt can be striped or chequered. Textures, as the ones shown in Figure 1.1, can be vividly described by humans using a variety of words in the English language. In this thesis we study the problem of generating meaningful and visually informative *texture descriptions* automatically by recognising a *combination of perceptual properties* for each texture. By addressing this challenge we add a novel dimension to texture recognition, aiming at *describing generic texture patterns* instead of focusing on texture identification or material recognition as commonly done. Furthermore, our goal is to solve this problem in *realistic settings*, that is, without making restrictive assumptions on the conditions in which images are captured, for example, under controlled illumination, viewing angle or scale. To this end, we consider images collected *in the wild*, randomly sampling them from the Internet. However, we limited the conditions to the one that occur most frequently,

Figure 1.1: We address the problem of *describing textures* by associating to them a collection of attributes. Our goal is to understand and generate automatically human-centric descriptions such as the examples above.

and avoided images in conditions unlikely to occur, or the ones that would make the recognition task challenging for humans: extreme light conditions such as insufficient light or high specularities, highly reflective surfaces, perspective of extreme, unrealistic distortions. For some of our experiments, we require that textures cover almost the entire image ($\geq$ 90% of area). For the most recent experiments, however, we allow textures to appear *in clutter*, by removing the common simplifying assumption in texture recognition that a single texture or pattern covers an entire image. We allow textures to cover regions of image of arbitrary shape, as long as these segments are connected, and sufficiently large such that they capture enough information in order to be recognizable. By removing this constraint, that is, allowing textures to appear as segments of arbitrary shape and size, we bring texture recognition closer to real-life applications.

Some examples of the texture attributes and materials addressed in this thesis,

Figure 1.2: **Texture recognition in the wild and in clutter**. Examples of top retrieved texture segments by attributes (top two rows) and materials (bottom) in the Open Surfaces dataset as recognised by one of our systems.

collected in the wild and in clutter, are shown in Figure 1.2. The figure shows also the predicted labels for these images, as recognized by one of the systems developed in this work.

The first step in achieving our aim is to select a rich vocabulary of words or attributes that can be used to describe a wide range of textures; this vocabulary should be compact, yet sufficiently large and the terms should be visually meaningful. The second step is to collect a large database of images to exemplify the words in the proposed vocabulary, in order to learn to describe textures, as well as to evaluate and compare algorithms. The third step is to develop a system capable to recognize a term or a list of terms, from a pre-defined vocabulary, which could be used as a description for the contents of a texture image. In order to do so, we propose novel texture representations that, compared to existing ones, are better

able of handling the huge visual variability of texture attributes.

1.2 Motivation and Applications

The primary motivation of developing texture attributes is to enrich the space of visual attributes that can be understood by computer vision systems. Recently, several contributions have demonstrated the power of attributes in *search*, by understanding complex queries, in *learning*, by porting textual information back to the visual domain, an in image *description*, by producing rich accounts of the content of images. Textural properties have an important role in these descriptions, particularly for objects that are best qualified by a pattern, such as textile, or the wings of birds or butterflies. Nevertheless, the attributes of textures have been investigated only tangentially. Our work aims at filling this gap by proposing a rich set of texture attributes inspired by the psychological literature [Bhushan et al., 1997].

A second motivation for our work is technical: work on recognizing textures motivated the development of orderless image representations. While these representations come natural for textures, they were shown to be applicable on a wide variety of tasks in computer vision, such as object or scene understanding. However, with the recent progress in deep learning, these classical methods have been replaced by the modern generation of deep convolutional neural networks. Therefore, we are interested in understanding how we could reuse methods from the classical texture representations in the context of deep learning, or at least how to draw inspiration from these methods.

Textures are often associated in computer vision to the problem of recognizing an instance of a surface or categories of materials. Perceptual properties such as attributes are orthogonal to these as: (i) they are subjective rather than objective; (ii) they are not exclusive but can be *combined* to describe each texture occurrence; and (iii) they are not intrinsically invariant to viewpoint and illumination, but only

invariant to the extent that human perception is (for example the same material may appear "cracked" or "smooth" depending on the illumination). This poses interesting challenges in modelling and recognizing such texture properties.

We are also motivated by the practical aspects of building computer vision systems capable of recognizing texture attributes. These have in fact the potential of enabling new human-centric applications, or of improving existing ones, by allowing computers to understand more nuanced and detailed properties of objects. For example, in this thesis we demonstrate applications of texture attributes to the automatic description of items in online catalogues of fabrics, as well as to the retrieval of such items based on their descriptions.

1.3 Challenges

A first challenge in developing a system capable of describing textures in terms of their visual properties is to identify a *vocabulary of describable texture attributes* capable of generating informative descriptions of a broad variety of textures. The list of selected terms needs to be informative, visually meaningful and broad, yet compact. The main questions that we ask are then: which are the words most frequently used in describing the visual appearance of textures? Can we identify a minimal set of such words that can satisfactorily describe a wide variety of textures?

Having chosen the texture attributes vocabulary, the second challenge is to collect in a *cost-effective manner* a reasonably large set of example images representative for each of the texture attributes. This data should be sufficiently large to learn and evaluate reliable models of the attributes that can generate coherent descriptions of the textures involving combinations of different attributes. Questions related to this goal include: how can we mitigate the costs of collecting annotations for a large number of images as well as attributes? Can we reduce the cost of collecting annotations by avoiding to gather uninformative ones? How can we obtain reliable

labels for the example images given that the attributes are subjective?

These questions enumerated so far are answered in Chapter 3, where we induce a texture attribute vocabulary from psychological studies, and where we explain how a novel dataset of describable texture attributes was collected and annotated.

A second challenge is to develop methods to recognise texture attributes automatically in images. The difficulty of this task can be appreciated by considering the huge visual variability that characterise visual attributes, as the same attribute can apply to many different "styles" of textures. This should be contrasted to standard problems in texture analysis such as instance identification and material recognition where variability is more limited. Recognizing texture instances, particularly in controlled conditions, is not a very challenging task, as seen from the near-perfect accuracies reported in the literature (and corroborated by the experiments in Section 5.2.1). Moving from *instances* to *material categories* adds more complexity as it introduces intra-class variations. A representative example is the KTH-TIPS2-b dataset, which contains just four different instances for each of a number of material categories. This variability can be appreciated in Figure 1.3, where four wool samples have very different appearance despite being photographed under the same viewpoint and illumination. The KTH-TIPS2-b benchmark is still not entirely saturated by current recognition methods.

Texture attributes, however, are likely to have an even greater visual variability than materials. In order to develop a generally applicable texture lexicon, one needs in fact to recognise a huge diversity of pattern types, from regular (*e.g.* chequered, polka-dotted, honeycombed) to stochastic (flecked, sprinkled,blotchy, smeared), and from micro-textures (porous, woven, pitted) to larger shape variations (bumpy, pleated). Furthermore, while materials may be correlated with certain texture attributes, recognizing attributes is not the same as recognizing materials. For example, a texture like *polka-dotted* can be painted on top of a wide diversity of support

Figure 1.3: Intra class variation, under controlled conditions. These images from KTH-TIPS2-b dataset depict four samples of wool captured at the same scale, viewing angle and illumination conditions.

materials such as fabric, plastic, or leather, as shown in Figure 1.4.

Recognizing texture attributes or materials of textures *in the wild* increases the recognition complexity significantly. Effects such as transparency or translucency of materials such as glass, plastic, wax, or the reflectivity and specularity of materials such as metal and plastic, cause the appearance of materials to depend on the environment surrounding them. Furthermore, materials need to be recognised independently of the objects that they form since the same type of objects can be made of different materials (*e.g.* a toy car can be made of metal, plastic or wood [Adelson, 2001; Liu et al., 2010]) and different objects can be made of the same material (*e.g.* a toy car and bottle can both be made of plastic, yet they look very different). The appearance of materials depends significantly on the objects that they form, and it is sometimes very challenging to factor materials from objects in images. For attributes, furthermore, the polka dots of Figure 1.4 reveal some of the complexity of visual attributes when imaged in the wild, on real objects: the size of the dots varies, even within the same texture instance; the layout of the texture (the spatial arrangement of the dots) and style (their colour) can change; occlusions, caused by the shape of the textured object or by other ones, mean that the texture region has a highly variable shape. By design, the images collected for our benchmarks are captured in natural light conditions (indoor or outdoor), as well as standard artificial light – but the light source parameters are not known upfront, and in general,

Figure 1.4: Intra-class variation for textures collected in the wild. The same texture can appear on various surfaces – leather, wall, fabric, rubber; the colour and size of the texture elements may differ, even within the same texture. Parts of the texture may not be visible, due to self-occlusions or occlusions by other objects.

multiple images of the same texture or material are not available, under varying light conditions.

When textures are also observed *in clutter*, there is one more layer of complexity as images must be segmented into textured regions before these can be assigned a label. Potential difficulties in segmentation may be caused by the scale of the texture, which can lead to over-segmentation due to the failure of algorithms to group texture elements (*e.g.* the squares in a chequered pattern); in turn, this may lead to an incorrect labelling of the pixels in the segments.

1.4 Thesis Outline

In this thesis we are introducing a novel dimension to texture analysis, by recognizing a large set of *perceptual attributes* of textures and generating texture descriptions, instead of recognizing textures by material or instances. Furthermore, we are ad-

dressing the problem of recognition in the wild and in clutter, removing constraints which may otherwise limit the use of texture attribute and material recognition in practical applications.

We are reviewing the related literature in **Chapter 2**, covering the prior work on perceptual properties of textures. In Section 2.2, we provide a detailed overview of specific benchmarks, addressing both instance recognition as well as category recognition, for textures and materials. We will cover early benchmarks, such as Brodatz dataset, up to recent and more challenging ones, such as Flickr Material Dataset (FMD). In terms of texture representations, in Section 2.3 we present classical texture representations, with an emphasis on local descriptors.

In Chapter 3 we introduce a rich vocabulary of 47 *describable texture attributes* inspired by the psychological literature [Bhushan et al., 1997] that capture a wide variety of visual properties of textures. We also introduce a corresponding *Describable Texture Dataset* consisting of 5,640 texture images *jointly* annotated with the 47 attributes (Section 3.1). In an effort to support directly real world applications, and inspired by datasets such as *ImageNet* [Deng et al., 2009] and the *Flickr Material Dataset* (FMD) [Sharan et al., 2009], our images are captured "in the wild" by downloading them from the Internet rather than collecting them in a laboratory.

We also address the practical issue of crowd-sourcing this large set of joint annotations efficiently accounting for the co-occurrence statistics of attributes and for the appearance of the textures. In Section 3.1.1 we are describing the annotation process in detail, covering the tools developed for this purpose and design decisions.

Although images in DTD and FMD are collected "in the wild", the textures always fill the view of the camera. To remove this last-standing limitation, we build on the *Open Surfaces* (OS) dataset that was recently introduced by Bell et al., 2013 in the computer graphics community, which we describe in more detail in Section 3.2. OS comprises of a large number of images, with a number of high-

quality texture/material segments. Many of these segments are annotated with additional properties such as the material name, the viewpoint, the BRDF, and the object class. Here we propose to use the material annotations in order to create a benchmark to evaluate material recognition in clutter. We also augment the OS dataset with some of the 47 attributes from DTD in order to evaluate in clutter the recognition of texture attributes as well.

In **Chapter 4**, we cover in more detail image representations, addressing local descriptors (Section 4.1), handcrafted, such as filterbanks or Local Binary Patterns, and proposing learned ones, like the outputs of convolutional layers of a Convolutional Neural Network (CNN). We are also comparing pooling encoders in Section 4.2, both orderless, like Bag of Visual Words, VLAD, Fisher Vector, as well as order-sensitive, like CNN-based pooling, using the fully connected layers of the CNN, or Spatial Pyramid Pooling.

In this context, we are introducing FV-CNN, a texture representation based on Fisher Vector (FV) encoding, and inspired by the recent advances in Convolutional Neural Networks (CNNs). CNNs have emerged recently as the new state-of-the-art for recognition, as demonstrated by remarkable results in image classification [Krizhevsky et al., 2012], detection [Girshick et al., 2014] and segmentation [Hariharan et al., 2014] on a number of widely used benchmarks. Key to their success is the ability to leverage large *labelled* datasets to learn increasingly complex transformations of the input to capture invariances. Importantly, CNNs pre-trained on such large datasets have been shown [Oquab et al., 2014; Chatfield et al., 2014; Girshick et al., 2014] to contain general-purpose feature extractors, transferable to many other domains.

The idea of FV-CNN is to interpret the convolutional layers of a CNN as a filter bank that extracts local image descriptors densely, and build out of these a representation using FV as an orderless pooling mechanism, similarly to bag-of-

words approaches. Although the suggested approach is simple, it is remarkably flexible and effective. First, pooling is orderless and multi-scale, hence suitable for textures. Second, any image size can be processed by convolutional layers, avoiding costly resizing operations. Third, convolutional filters, pooled by FV-CNN, are shown to transfer more easily than fully-connected ones even without fine-tuning. While other authors have recently proposed alternative pooling strategies for CNNs [He et al., 2014; Gong et al., 2014], we show that our method is more natural, faster and often significantly more accurate.

In Chapter 5 we provide a *thorough evaluation of the proposed descriptors*, on a variety of benchmarks, from attributes to object categories and from textures to scenes. For texture analysis, we evaluate material and describable attributes recognition and segmentation in the wild and in clutter using our new DTD and Open Surfaces datasets (Section 3.2), as well as in a large number of standard benchmarks. When used with linear SVMs, FV-CNN improves the state of the art on texture recognition by a significant margin. We also show that the proposed descriptor can be used for other recognition tasks, like scenes, objects and even fine-grained classification. What is remarkable is that on MIT Indoor Scenes dataset, FV-CNN does not only significantly outperform the current state of the art of Zhou et al., 2014, but also removes entirely the domain-specific advantage of the CNNs trained on scene data, showing that FV-CNN is better at domain transfer.

In Chapter 6, we extend the evaluation to the problem of texture segmentation and recognition. We evaluate the proposed descriptors on image regions obtained from off-the-shelf segmentation algorithms, to show that using better region classifiers could contribute to further refining image segmentation.

In Chapter 7 we explore potential applications of the proposed describable attributes. We introduce a *semantic*, *low-dimensionality* mid-level descriptor, obtained as scores of the 47 linear classifiers learned on DTD. These can serve a

complementary role for recognition and description in domains where the material is not-important or is known ahead of time, such as fabrics or wallpapers. However, can these attributes improve other texture analysis tasks such as material recognition? We answer this question in the affirmative in a series of experiments on the challenging FMD and KTH-TIPS2-b datasets. We show that these describable descriptors, when used as features, can give significant boosts in accuracy. Furthermore, these attributes are "human readable" and can serve as intuitive dimensions to explore large collections of texture patterns – for example product catalogues (wallpapers or bedding sets) or material datasets. We present several such visualizations in this chapter and in the Appendix.

We conclude this thesis, with a summary and suggestions for future work in Chapter 8.

1.5 Publications

This section gives a list of the publications that contain the content of this thesis.

Published work

- **CVPR 2014 [Cimpoi et al., 2014]** – This paper introduces the problem of recognizing describable texture properties, and DTD, a dataset suitable for the problem, with a detailed description of the collection framework (presented in Chapter 3). In the paper we also proposed a state-of-the-art descriptor for texture and material classification, based on Fisher Vector pooling of dense SIFT features, and deep convolutional features.

- **CVPR 2015 [Cimpoi et al., 2015]** – This paper proposes a novel texture descriptor, by pooling the activations of convolutional layers of deep convolutional neural networks, using Fisher Vector encoding (Chapter 4), and shows its application to recognizing textures (and materials) in the wild and clut-

ter. We evaluated the proposed descriptor on standard benchmarks, as well as Open Surfaces dataset, which we extended with semantic attributes.

Work under review

- IJCV 2015 – This paper provides an in depth analysis of parameters and variants of the new descriptor introduced in Cimpoi et al., 2015, as well as a thorough evaluation for several benchmarks: texture, material, object, scene recognition, as well as fine-grained categorization.

Chapter 2

Literature Review

This chapter reviews the literature most closely related to our work on texture analysis. We will cover the prior work on perceptual properties of textures in Section 2.1. Then, in Section 2.2 we will introduce the most notable benchmarks for texture *instance* recognition as well as *category* recognition. The new benchmarks we are proposing, Describable Texture Dataset and OpenSurfaces dataset, with the attribute extension are presented in Chapter 3. 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; Ojala et al., 2002; Varma and Zisserman, 2003; Varma and Zisserman, 2005; Sharan et al., 2013; Schwartz and Nishino, 2013], segmentation [Manjunath and Chellappa, 1991; Jain and Farrokhnia, 1991; Chaudhuri and Sarkar, 1995; 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]. This thesis is mostly concerned with the problem on texture recognition, discussed in Section 2.1 for perceptual properties, and in Section 2.2 for the recognition of identities and materials. The latter

section reviews in some detail existing texture benchmark datasets, both because they are related to our own development of a novel benchmark and because they will be used in the evaluation of texture representations.

The chapter concludes with Section 2.3 presenting an overview of texture representation and recognition methods. In this section we focus on the methods achieving the state-of-the-art, and leave the details about local descriptors for Chapter 4, Section 4.1.1, as well as pooling encoders, presented in Section 4.2, as they are closely related to the proposed method.

2.1 Perceptual Properties of Textures

In the analysis of visual textures, the focus was predominantly on the recognition of *instances* and *materials* [Adelson, 2001], but perceptual properties such as *describable texture attributes* [Ferrari and Zisserman, 2007; Matthews et al., 2013] have received increasing attention. Describable attributes include properties such as "lined" or "crosshatched" that are not necessarily mapped to higher-level semantic categories such as objects, parts, or materials. While the literature on describable texture attributes is much smaller than the one on material recognition, the interest in such attributes is both scientific – *i.e.* as a way of developing a deeper understanding of textural patterns – and applicative – *e.g.* for e-commerce, assisted design, computer graphics, and human interfaces.

One of the key contributions of this thesis is to 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. This work is motivated mainly by the work of Bhushan, Rao and Lohse [Rao and Lohse, 1996; 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 thesis is the challenge of estimating such properties from images automatically. In particular, in Bhushan et al., 1997, the authors identified a set of words sufficient to describing a wide variety of texture patterns; the same set of words was used to bootstrap DTD.

While recent work in computer vision has been focused on texture identification and material recognition, notable contributions to the recognition of perceptual properties exist. Texture attributes have an important role in describing objects, particularly for those that are best characterised by a pattern, such as items of clothing and parts of animals such as birds. Remarkably, the first work on modern visual attributes by Ferrari and Zisserman, 2007 focused on the recognition of a few perceptual properties, that describe textures. In this paper, the authors propose a generative model for visual attributes, which may be unary – like colours, basic textures (sand, grainy), shape – or binary, when the basic element is composed of two segments, such as the black and white stripes in the case of a zebra. While in texture and material recognition benchmarks, the classes are disjoint, multiple attributes could be used to describe a single image. The idea of multi-label learning originated from text classification [McCallum, 1999], and it was adopted for vision applications as well [**zhou2007multi**].

However, the problem of recognizing attributes was formulated in the context of objects. In [Farhadi et al., 2009], the authors propose shifting the goal of recognition, from naming to describing, which comes with several advantages, including: recognizing unusual properties of familiar objects, describe unknown objects, and enables learning how to recognize new objects, with very few visual examples, or even none. A list of 64 attributes was proposed, and images from Pascal VOC2008 were labelled with these attributes, using Amazon Mechanical Turk. Besides using semantic attributes, which may describe presence or absence of parts (*e.g. has wheel, no beak*), shape or material (*furry*), a set of discriminative attributes are used, to

help better separate between classes that share the same set of semantic attributes (*e.g. cats and dogs are both four-legged and furry*). In [Farhadi et al., 2010], also in the context of objects, it is shown that using attributes contributes to localizing and describing familiar and unfamiliar objects better, compared to a baseline that relies only on basic category detectors. Attributes were also used for "zero-shot" transfer learning, [Lampert et al., 2009], to overcome the lack of training examples for certain novel categories. Attributes could be reliably used to predict new classes, based on their attribute representation, without the need for retraining.

Later work, such as Berg et al., 2010 mined visual attributes from textual descriptions of images on the Internet, in an attempt to discover relevant attributes, for a large number of categories, without the need of manual labelling. The discovered attributes include mostly colour, shape and style properties, but also texture attributes, such as "striped" or "plaid".

The work of Parikh and Grauman, 2011 proposes the use of relative attributes, as opposed to previous work, which considered attributes binary, indicating presence or absence of certain properties. Relative attributes seem more natural and less restrictive and enable more meaningful descriptions, by expressing the strength of a property with respect to a reference example. From the technical perspective, learning relative attributes implies learning a ranking [Joachims, 2002] of images, based on the strength of the properties, and the authors show experimentally that learning relative attributes gives better results compared to learning binary attributes.

Relative attributes were used in a feedback mechanism, for guiding and refining retrieval of shoes and clothing items [Kovashka et al., 2012]. A set of ranking functions are learnt offline, and at query time, the relevance function re-ranks the pool of images given the constraints provided by the user. This approach allows the users to iteratively refine the results, starting from a generic query. While the predictions improve with the amount of feedback provided, relative attributes

feedback shows more significant gains per unit of feedback.

Nevertheless, so far the attributes of textures have been investigated only tangentially, as most of the work on visual attributes has relegated describable texture properties to a marginal role [Farhadi et al., 2009; Farhadi et al., 2010; Parikh and Grauman, 2011], focusing on attributes of objects.

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 renders of 3D reconstructions of real materials) as well as a semantic similarity matrix obtained from human similarity judgements. The work most related to ours is probably the one of Matthews et al., 2013 that analysed images in the Outex dataset [Ojala et al., 2002] using a subset of the texture attributes that we consider. Our dataset 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 show the transferability of the texture attributes to real-world applications, including material recognition in the wild and in clutter.

2.2 Texture Instances and Material Categories

Texture recognition has been explored in a variety of scenarios. A first axis of variability is whether the focus is on recognising specific texture or material samples (instance recognition) or whether the goal is to classify textures based on material (category recognition). A second important axis of variability is whether textures are imaged in controlled conditions, as in benchmarks such as CuRET [Dana et al., 1999] or KTH-TIPS [Caputo et al., 2005], or, more recently, whether textures are collected "in the wild", as evaluated for example by FMD [Sharan et al., 2009].

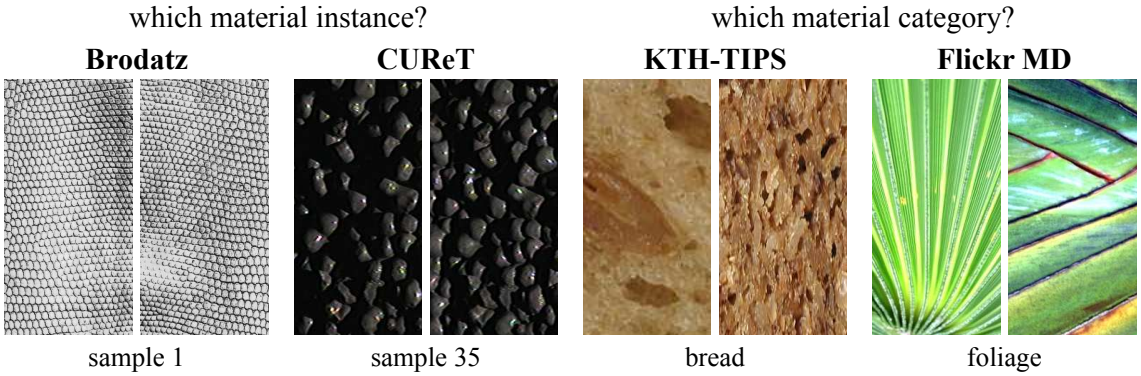


Figure 2.1: Datasets such as Brodatz Brodatz, 1966 and CUReT Dana et al., 1999 (left) addressed the problem of material instance identification and others such as. KTH-T2b Hayman et al., 2004 and FMD Sharan et al., 2009 (right) the one of material category recognition. Our DTD dataset addresses a very different problem: the one of describing a pattern using intuitive attributes (Figure 1.1).

2.2.1 Texture and Material Instances

Datasets like *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, in the sense that the categories are instances of the same object, but 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].

One of the first proposed texture datasets is **Brodatz** [Brodatz, 1966], which consists of 111 images which are cut into 9 non-overlapping patches of equal size, giving 999 image samples, 9 per class. These textures are greyscale images and don't present any viewpoint and illumination variations for the samples that belong to the same instance, that is, extracted from the same photograph in the album. The small number of samples and lack of inter-class variation make this dataset a

	Size			Condition			Content			Instances /
Dataset	Images	Classes	Splits	Wild	Clutter	Controlled	Attr.	Mat.	Obj.	Categories
Brodatz	999	111	–			✓			✓	I
CUReT	5612	61	10			✓		✓		I
UIUC	1000	25	10			✓		✓		I
UMD	1000	25	10	✓					✓	I
KTH	810	11	10			✓		✓		I
Outex	–	–	–			✓		✓	✓	I
Drexel	∼40000	20	–			✓		✓		I
ALOT	25000	250	10			✓		✓		I
FMD	1000	10	14	✓				✓		C
KTH-T2b	4752	11				✓		✓		C
DTD	5640	47	10	✓			✓			C
OS	10422	22	1	✓	✓			✓		C
OSA	1720	10	1	✓	✓		✓			C

Table 2.1: 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. Note that Outex is a meta-collection of textures spanning different to problems.

simple one, fact confirmed also by the high accuracy achieved on this set.

The most popular and widely used texture database is **CUReT** [Dana et al., 1999], which contains 61 material classes, collected under 200 different combinations of viewing and illumination directions. Only a subset of 92 images per class, containing cropped central region, are commonly used for reporting results, the other samples being filtered out due to extreme viewpoint conditions. Compared to Brodatz, the novelty in CUReT is using colour images, representing common material surfaces, and it contains a larger number of samples per class.

KTH-TIPS (Textures under varying Illumination Pose and Scale) database was proposed [Hayman et al., 2004] to compensate the lack of scaling effects in the CUReT database, and was designed aiming to supplement CUReT. KTH-TIPS contains 10 material classes, a subset of CUReT materials, and there are 81 images for each sample, representing all the combinations of 9 scales, 3 different illumination directions (front, side and top) and three object poses.

Another dataset used for texture classification is **ALOT** (Amsterdam Library

of Textures) [Burghouts and Geusebroek, 2009], which is a colour image collection of 250 rough textures, similar to CUReT, and although it exposes only a half of the view-illumination directions per material available in CUReT, ALOT expands the number of materials approximately 5 times and also introduces mixtures of materials and improves upon image resolution and colour quality.

UIUC is a set consisting of 1000 uncalibrated, greyscale images, with 40 samples for each of the 25 texture categories [Lazebnik et al., 2005]. The resolution of the samples is 640 x 480. The textures considered have the appearance given by albedo variation (wood or marble), 3D shape, in the case of fur, and even a mixture of the two, for bricks. The images were captured under uncontrolled illumination, and additional sources of variation, such as non-planarity of the surface, non-rigid deformations between different samples of the same class and viewpoint-dependent appearance variations, were deliberately introduced during acquisition.

A similar set, **UMD** [Xu et al., 2009], contains 1000 greyscale images, but the textures are non-traditional, including fruit, shelves of bottles, plants and floor textures. The higher resolution (1280 x 960) was needed in order to prove that MFS (multi-fractal spectrum) texture signature was invariant to local linear illumination, which has been obtained for the continuous case, assuming very small ratio of a measurement function to image resolution. The images in this dataset were captured "in the wild", in uncontrolled illumination and viewing angle.

2.2.2 Material Recognition

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., 2005] 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").

KTH-TIPS was extended to material recognition in **KTH-TIPS2** [Mallikarjuna et al., 2005], collected following a similar procedure as for KTH-TIPS, with some differences regarding scale range and illumination. The same three poses (frontal, rotated 22.5° left and right) were kept, but an additional illumination condition was introduced, by using fluorescent lights. There are two versions of KTH-TIPS2 dataset: KTH-TIPS2-a, in which, for four of the samples the images from the initial database were used, thus having only 72 images per sample, and the completed KTH-TIPS2-b version, which had 108 images for all the 11 samples.

Most of the datasets reviewed so far make the simplifying assumption that textures fill images, and often there is limited intra-class variability, due to *single or limited number of instances*, captured under controlled scale, view-angle and illumination. Thus, they are not representative of the significantly harder 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 focus the *Flickr Material Database* (FMD) [Sharan et al., 2009]. FMD samples just one viewpoint and illumination per object, but contains many different objects instances grouped in several different materials classes. FMD consists of 10 material classes: wood, paper, stone, metal, plastic, glass, fabric, leather, water and foliage, having 100 images per category. It is more challenging than the existing texture databases, because it contains images belonging to high level classes, which show high inter-class variation in appearance and object geometry. This dataset was collected from Flickr, and was originally designed for studying visual perception of materials. Along with KTH-T2b, this is the first *proper* material dataset, other datasets containing images of a single texture instance per class, photographed under various viewing conditions; a crucial difference with KTH-T2b is that most texture instances in FMD appear only once in the dataset, under a single viewpoint and illumination.

Most of these datasets discussed so far are close to saturation with state-of-the-art classification accuracy 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 (using nearest-neighbour methods) to classify textures, significantly improving on other methods. Table 2.1 provides a summary of the nature and size of various texture datasets that are used throughout our experiments.

2.3 Texture Representations

In order to recognise textures automatically it is necessary to construct mathematical descriptions or representations of texture images. Early representations built on Julesz's conjecture [Julesz, 1962] that two textures are perceptually indistinguishable if they have identical second order statistics. Later, this conjecture was disproven, and the notion of textons was introduced, to define primitive elements of texture [Julesz, 1981], like crossings, orientation elements and terminators.

Initially, texture recognition was considered primarily a 2D problem. The idea of consider natural texture recognition as a 3D problem was first presented in Leung and Malik, 1999, which introduces the notion of 3D textons. In this paper, 2D textons are defined as prototype vectors, which are cluster centres from filter responses, which is the first practical definition of textons. This definition is further extended by considering a stack of images of the same texture, from different lighting and viewing directions, for which the filter responses are concatenated, and the K-means cluster centres are called 3D textons. These 3D textons are useful in capturing the 3D nature of the surface, shadows, occlusions, while the models from the traditional methods would be able to capture only surfaces that look flat, due to the assumptions that the surfaces are painted with a Lambertian material.

A similar method, but using 2D textons was proposed by Cula and Dana, 2001,

Figure 2.2: **Instance recognition benchmarks.** We show some representative samples from the most frequently used texture / material recognition benchmarks. For these datasets, the common characteristic of this datasets is that one instance of a given material or surface is photographed under controlled conditions. For example, two different leaves would be different classes in these datasets.

based on BTF (bidirectional texture function) [Dana and Nayar, 1998]. In this approach, during dictionary generation, the filter responses were grouped by scale, and clustered without being concatenated. This way, there were multiple models for each texture category, depending on viewing and illumination conditions. The authors also developed a method to reduce the number of models, using PCA for projecting the training and test histograms into a lower dimensional space.

Figure 2.3: **Material category recognition benchmarks.** We show some representative samples from KTH-TIPS2-b and Flickr Material Dataset (DTD). The former is collected in controlled conditions, but there are four "samples" photographed for each category. For the latter, each image represents a different instance of the category, captured in unknown and unconstrained illumination and viewpoint.

A texture is characterised in general by the arrangement of local patterns, which was captured in early works [Leung and Malik, 2001; Varma and Zisserman, 2003] by the distribution of local "filter bank" responses. These filter banks were designed to capture edges, spots and bars at different scales and orientations. Typical combinations of the filter responses, identified by vector quantization, were used as the computational basis of the "textons" proposed by Julesz [Julesz and Bergen, Jul-Aug 1983]. Texton distributions were the early versions of "bag-of-words" rep-

resentations, a dominant approach in recognition in the early 2000s.

The filter bank proposed by Leung and Malik [Leung and Malik, 2001] is a multi-scale, multi-orientation filter bank, consisting of 48 filters. These are bar filters – two derivative of Gaussian, at six orientations and three scales, and spot filters, 8 Laplacian of Gaussian and 4 Gaussian filters. S (Schmid) filter bank [Schmid, 2001], consists of 13 rotation invariant filters, which combine scale and frequency.

The MR8 filter bank [Varma and Zisserman, 2002; Varma and Zisserman, 2005; Geusebroek et al., 2003] consists of 38 filters, but only 8 responses are recorded. These 38 filters are: a bar filter and an edge filter, at three scales and six orientations, but only the maximum response across orientations is used, resulting in 6 responses; the other two filters are a Gaussian and a Laplacian of Gaussian.

A simple, yet very efficient descriptor was proposed by Ojala et al., 2002. The descriptor, based on local binary patterns (LBP), at multiple resolutions, achieves greyscale and rotation invariance. The operator, used to obtain "uniform" binary patterns, is designed with two parameters, one that controls the spatial quantization of the angular space and one which controls the spatial resolution. Although initially designed for textures, the proposed descriptor was successfully extended and applied to textures [Sulc and Matas, 2014] and other recognition tasks as well.

The VZ-Joint classifier [Varma and Zisserman, 2009] surpassed existing state-of-the-art classifiers, based on filter responses, by considering joint distribution of pixel intensities in a neighbourhood (a square patch centred at the pixel). The authors evaluated two versions of the Joint classifier – a neighbourhood classifier and an MRF classifier. In the case of the neighbourhood classifier, the central pixel of the patch is discarded, and only the neighbourhood is used for classification. The MRF classifier, on the other hand, models explicitly the joint distribution of the central pixel and its neighbours.

A detailed evaluation was presented in Mikolajczyk et al., 2005, comparing vari-

ous local descriptors for local interest regions, and how these depend on the interest region detector [Mikolajczyk and Schmid, 2005]. This work compares the descriptors in the context of matching two images of the same *scene*, under variations: rotation, zoom, viewpoint change. In Mikolajczyk et al., 2005, the authors proposed an extension of SIFT [Lowe, 1999], designed to increase its robustness and distinctiveness. This descriptor, called gradient location and orientation histogram (GLOH) is computed as a 272 bin histogram, using 17 location bins, placed angularly, at three radii, and gradient orientations being quantized in 16 bins. This is reduced via PCA to 128 dimensions.

The **DAISY** descriptor [Tola et al., 2008] was inspired by SIFT and GLOH, can be very efficiently computed. At each pixel location, the DAISY descriptor consists of values of orientation maps, for different scales, computed on a circular grid, as opposed to the regular one, used in SIFT. There were several improvements to DAISY descriptor, like robust normalization, dimension reduction [Winder et al., 2009; Brown et al., 2011], resulting in a compact and highly efficient local descriptor. Simonyan et al., 2012, propose using convex optimisation for learning the pooling regions and reducing the dimensionality of the DAISY descriptor.

Liu et al., 2010 proposes a method to exploit the correlation between reflectance properties and the material's high-level category, in order to classify images containing materials which belong to ten categories, using FMD [Sharan et al., 2009] as benchmark, and the features used for discrimination include both low and mid-level features, which are combined under a Bayesian generative framework, using augmented Latent Dirichlet Allocation (aLDA). The results were improved by using multiple features with a discriminative SVM classifier [Sharan et al., 2013], to 57.1% accuracy.

Comparable results for FMD are reported, obtained using Basic Image Features (BIFs), and a *training-free* framework, based on Nearest Neighbour Classifier and

Nearest Mean Classifier (NMC) [Timofte and Van Gool, 2012]. The method also obtains top classification results on standard benchmarks for texture classification (CUReT, UMD, UIUC), and notably, KTH-T2b, improving by 8% on the previous best.

Local higher-order statistics (LHS) vectors of Sharma et al., 2012 exploit the rich information of small (3x3) pixel neighbourhoods. A local differential vector is obtained by subtracting the (grey) value of a central pixel from the values of the other pixels from the 8-neighbourhood. Then, a Gaussian Mixture Model is used for soft quantization, and the image representation is obtained by averaging the Fisher score of all differential vectors.

The method proposed by Sifre and Mallat, 2013, consists of a deep convolution network, with wavelet filters and modulus non-linearities. Invariance to rotations and translations is achieved using the scattering transform operator. This architecture is similar to convolutional networks, with the difference that the filters are not learned, but they are scaled and rotated wavelets.

Ffirst [Sulc and Matas, 2014], is a computationally efficient descriptor, based on LBP. It uses LBP histogram Fourier features, for both sign and magnitude, and the two histograms are concatenated to obtain the descriptor. Multiscale invariance is obtained by using circular neighbourhoods with an exponentially growing radius, and a finer scaling. Surprisingly, the Ffirst descriptor achieves similar performance as the combination of Fisher Vector pooled SIFT and fully connected layer activations (DeCAF [Donahue et al., 2013]), as introduced in Cimpoi et al., 2014.

More recently, texture and material recognition methods have improved, due to new pooling schemes such as soft-assignment [Wang et al., 2010; Zhou et al., 2010; Liu et al., 2011] and Fisher Vectors (FVs) [Perronnin et al., 2010]. Until recently, FV with SIFT features [Lowe, 1999] as a local representation was the state-of-the-art method for recognition for objects and scenes; we ported some of these methods

to texture recognition in this thesis.

2.3.1 Deep Representations

Similarly to object recognition, most of the previous work on texture and material recognition focused on improving representations, from local features to encodings. The majority of these methods, which we summarised in Section 2.3, are mostly *handcrafted*, following the standard pipeline: computing local descriptors, then encoding them into a higher dimensional representation (Bag of Visual Words [Csurka et al., 2004; Sivic and Zisserman, 2003], VLAD [Jégou et al., 2010], Fisher Vector [Perronnin and Dance, 2007; Perronnin et al., 2010]). The local descriptors for texture recognition are in general responses of filter banks, distributions of patches or small neighbourhoods, or, more recently, LBP-based.

For object recognition tasks, *deep* representations, obtained using Convolutional Neural Networks [LeCun et al., 1989], were shown to outperform the standard (*shallow*) methods [Chatfield et al., 2011; Chatfield et al., 2014] on benchmarks such as PASCAL VOC Everingham et al., 2007 and ImageNet Deng et al., 2009. These representations, for both textures and objects, are fed to a classifier, which could be a linear SVM [Cortes and Vapnik, 1995], a random forest [Amit et al., 1997; Breiman, 2001], or in fact, any multi-class classifier. Since non-linear SVMs are costly to evaluate and train, they could be approximated by linear ones, using explicit feature maps [Vedaldi and Zisserman, 2010]. The idea was further extended to learning multiple kernels [Bach et al., 2004].

Recent state-of-the-art methods on object recognition using CNNs are still evolving as they require sophisticated implementations on GPU, the exploration of vast model spaces including variations in network architecture, and addressing large scale datasets and data augmentation [Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Chatfield et al., 2014; Sermanet et al., 2013; Jia, 2013].

As shown in Donahue et al., 2013 and Girshick et al., 2014, the first fully connected layer of a network trained on ImageNet can be used as a powerful descriptor on other datasets, including textures, as shown later in this thesis. Our work is related to Gong et al., 2014, which proposes using VLAD for pooling activations of fully connected layers of a CNN, densely sampled over the image. The key difference is that we are using the outputs of a convolutional layer, which is more natural and efficient, as we show later in Chapter 5, and does not incorporate long-range spatial information which is suitable for texture recognition. Another advantage of our method is also flexibility in selecting which and how many scales to use in representing an image, as well as the ability of quickly computing the representation for image subregions.

Conclusions. Texture is ubiquitous and provides important visual cues, in particular when shape and colour are not useful. There has been a lot of interest in textures in the vision community, which triggered the development of local descriptors and image representations. Although a number of these descriptors were developed to recognize textures captured under controlled conditions, realistic scenarios, with images collected "in the wild" are more difficult. More complex descriptors, as SIFT, and recently, CNNs are taking over. However, we should not discard these methods and simply replace them with CNNs, but rather draw inspiration from this prior work, to improve the way CNNs work, or design hybrid architectures, like FV-CNN, which will be introduced in the next Chapter.

Chapter 3

Describing Textures With Attributes

In this chapter we look at the problem of automatically describing texture patterns using a general-purpose vocabulary of interpretable texture properties, similarly to how we can vividly characterise the textures shown in Figure 1.1 using a variety of words in English. Our goal is to design algorithms capable of generating and understanding texture descriptions involving a *combination of describable attributes* for each texture. The aim is to fill a gap in the space of visual attributes that can be understood by computer vision systems for the purposes of material and texture recognition. Attributes have been 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 have an important role in these descriptions, particularly for objects that are best qualified by a pattern, such as a scarf or the wing of bird or of a butterfly. Nevertheless, the attributes of textures have been investigated only tangentially so far.

The first step in filling this gap is to introduce the **Describable Textures**

Dataset (DTD), a collection 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 Figure 3.1 and include words such as *banded*, *cobwebbed*, *freckled*, *knitted*, and *zigzagged*. Section 3.1 describes this data in more detail. Section 3.1.1 discusses the technical challenges 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.

3.1 Describable Textures Dataset

DTD investigates the problem of *texture description*, intended as the recognition of describable texture attributes. This problem differs from standard texture analysis tasks such as texture identification and material recognition (Section 2.2). 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*). Describable attributes can be *combined* to create rich descriptions (Figure 3.5; 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, they will be shown to be helpful in material recognition too (Section 7.1).

DTD contains **textures in the wild**, *i.e.* texture images extracted from the web rather than captured or generated in a controlled setting. In most of our images, texture fills the entire image – this was by design so that we can study the problem of texture description independently of texture segmentation. With 5,640 such im-

Figure 3.1: 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.

ages, this dataset aims at supporting real-world applications where 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 datasets for a representation of real-world variations, shortening the gap with applications. Furthermore, the invariance of describable attributes is

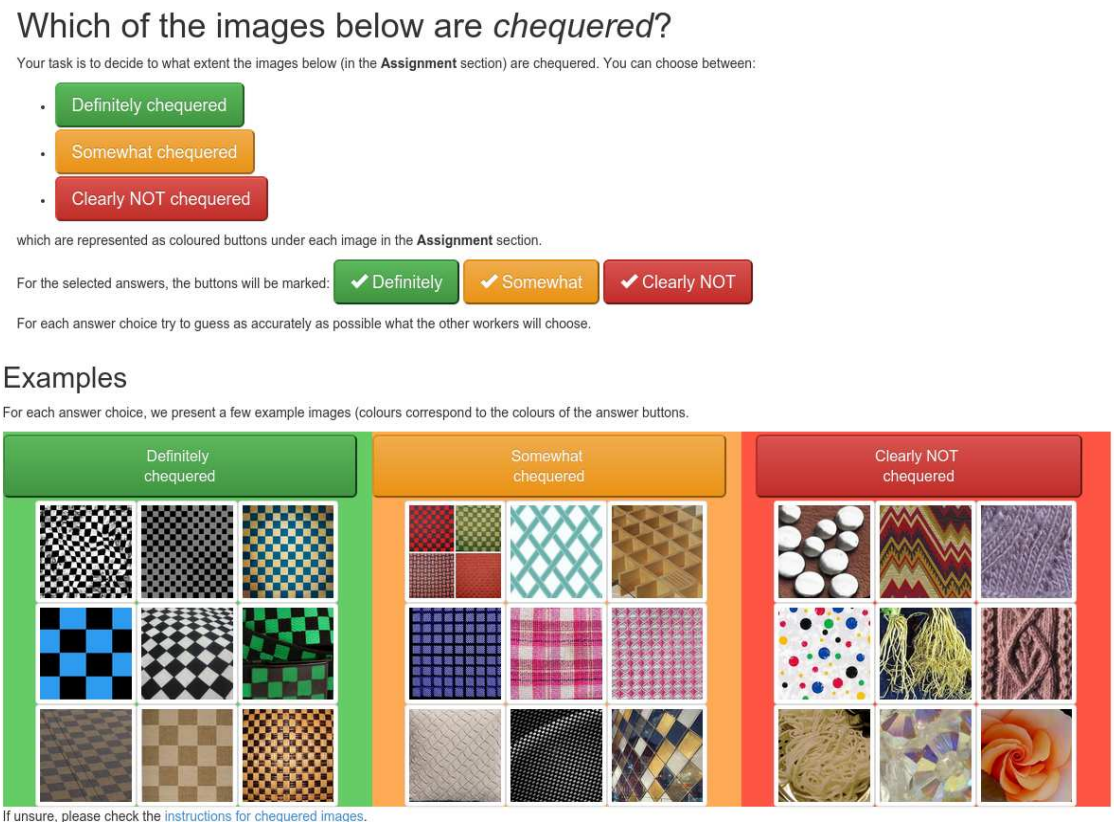


Figure 3.2: **Instructions for Amazon Mechanical Turk annotators.** For each of the 47 categories, the annotators were presented example images for each possible answer: *definitely belongs to the category, belongs somewhat* or *clearly the image does not belong* to the desired category. In addition to the example images, the dictionary definition and further explanations are provided. The choices are also colour coded, to match the answer option.

not an intrinsic property as for materials, but it reflects invariance in the human judgements, which should be captured empirically.

3.1.1 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.

3.1.2 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 smaller number of representative words. Our starting point is the list of 98 words identified by Bhushan, Rao and Lohse [Bhushan et al., 1997], which, while not being exhaustive, covers a wide variety of textures. Their seminal work aimed to achieve for texture recognition the same that colour words have achieved for describing colour 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 variability. Development of DTD is based on the work done at CLSP 2012 Summer Workshop [Blaschko et al., 2012]. The first iteration of this dataset consisted of 100 images for each of the 98 terms in the list, downloaded from Google image search and Flickr. However, after manual inspection, we discovered terms which refer to higher-level properties of texture, such as: *asymmetrical, complex, cyclical, discontinuous, harmonious, repetitive, rhythmic*. These terms describe the structure of the texture, and do not map to a certain texture category – both *chequered* and *honeycombed* could be regular, repetitive. Since these terms are too generic, and there are no representative images for them, we removed them from the list. We also noted that some of the terms are synonyms or visually similar *e.g. crinkled, wizened* and *wrinkled*, *coiled, corkscrewed* and *swirly*, or *gouged, furrowed* and *grooved*. For these terms, identified by similar image search results, and checking definitions in online dictionaries, we decided to merge the images corresponding to similar terms into one category. This resulted in a set of 47 words, illustrated in Figure 3.1.

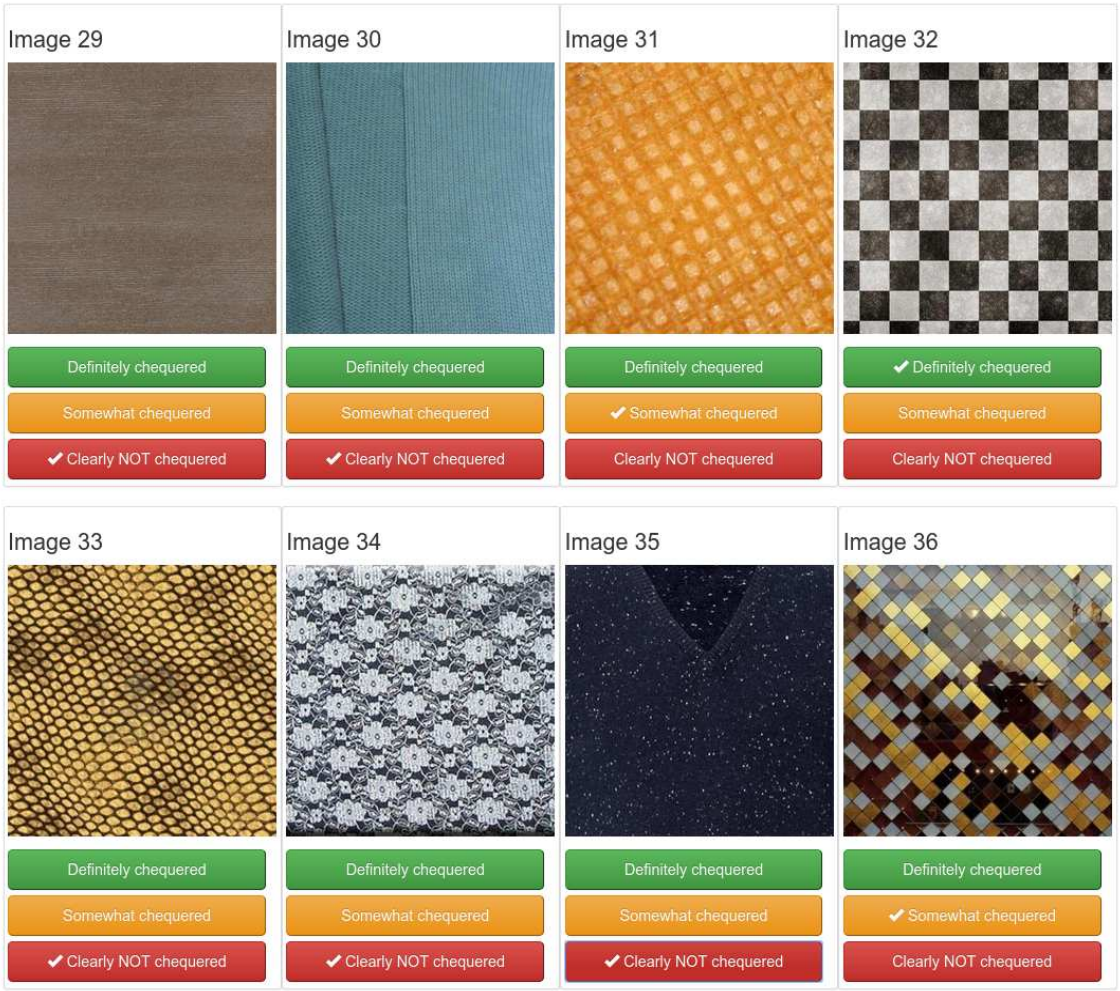


Figure 3.3: **Amazon Mechanical Turk Annotation Interface.** For each of the 47 categories, the annotators were presented a series of images, and were asked to indicate how much an image represents the category in question, in this example, chequered. The answers are shown with a checkmark.

3.1.3 Bootstrapping the Key Images

Given the 47 attributes, the next step was collecting a sufficient number (120) of example images representative of each attribute. A very large initial pool of about a hundred-thousand images was downloaded from Google and Flickr by entering the attributes and related terms as search queries. These queries included the category name, as well as augmentations of this query: *banded pattern, banded texture, chequered material, polka dots*. First, we prepared the images to be annotated on

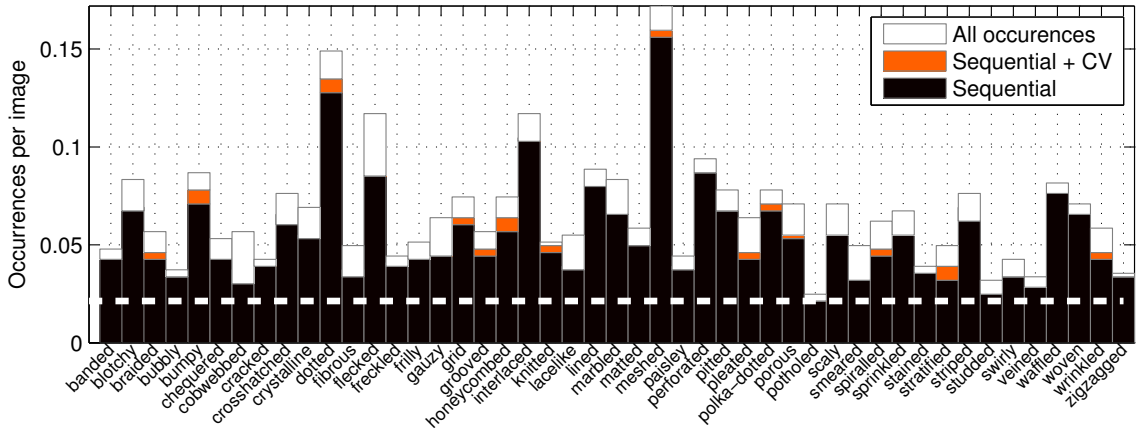


Figure 3.4: **Quality of joint sequential annotations.** Each bar shows the average number of occurrences of a given attribute in a DTD image. The horizontal dashed line corresponds to a frequency of 1/47, the minimum given the design of DTD (Section 3.1.1). The black portion of each bar is the amount of attributes discovered by the sequential procedure, using only 10 annotations per image (about one fifth of the effort required for exhaustive annotation). The orange portion shows the additional recall obtained by integrating CV in the process.

Amazon Mechanical Turk (AMT), by removing low resolution, poor quality, watermarked images, or images that were not almost entirely filled with a texture. Images with watermarks were removed directly from the queries, indicating Google image search to exclude paid stock image websites – because of watermark and copyright reasons. Also, queries contained parameters to return only images of size larger than 300×400 pixels.

Next, detailed annotation instructions were created for each of the 47 attributes, including a dictionary definition of each concept, from an online dictionary, and examples of correct and incorrect matches. Example instructions for the *chequered* class are shown in Figure 3.2. Votes from three to five AMT annotators were collected for the candidate images of each attribute, resulting in a short-list of about 200 highly-voted images was further manually checked by the authors to eliminate remaining errors.

The first iteration of annotations, done at CLSP Workshop, used three annotators per image, and The Mechanical Turk workers were presented with a set of

20 images in each batch. We noted that several images didn't meet the desired quality – watermarks, low resolution or not covering at least 90% of the image. We repeated the collection process, this time, filtering those images automatically, through query parameters, and through brief manual inspection before submitting them to Mechanical Turk.

We posted the images on Mechanical Turk in batches of 50 per assignment, and paid 0.05$ per assignment. This decision was motivated by the fact that Amazon charges 10% of the assignment cost, with a minimum fee of 0.005$, therefore we had to pay at least 5 cents of a dollar per task, in order to keep the amount of fees to 10% of the cost. The user interface for collecting the annotations, with example images and the corresponding answers is shown in Figure 3.3.

The work submitted by the annotators, called HITs (Human Intelligence Tasks) in the Amazon MTurk interface, can be approved by the requester, if it meets the desired quality, or rejected, otherwise. The annotators are paid only for the approved hits, and the number and percentage of approved work is an indicator of the annotator quality. The number of approved tasks or the percentage of approved work may be used as a qualifications to filter the annotators which are allowed to work on the submitted tasks. We selected workers with 90% approval rate and at least 100 approved HITs. We approved automatically the HITs, if the answers were in agreement with the majority vote for 70% of the images – a ratio which we found empirically to be sufficient to get enough useful annotations, and rejected the work otherwise. Images for which the votes were negative, that is, annotators indicated that the image does not reflect the term, or there was no clear majority vote (2 vs 3 answers) were added back to the pool of images to be annotated.

In the annotation process, we also experimented with repeating some of the images within the batch, rotated and with slight hue/saturation changes, to measure intra-annotator agreement. The goal of this approach was to filter out annotators

who solve the tasks randomly. This metric was computed as the percentage of duplicate annotations for which the answers were consistent. The disadvantages of this approach, shown to give similar results as using annotators' history, are increasing the cost per image with a percent equal to the percentage of duplicate images, as well as having unbalanced number of annotations per image. In our experiments, we used these extra annotations only for checking annotator's self-consistency.

Another experiment we considered was to treat the proposed attributes as relative, and to collect annotations for pairs of images. Although more accurate – because of larger number of annotations collected per image, this process would be very expensive. The number of pairwise comparisons would go quadratically with the number of images. Also, there are categories for which relative comparisons are not applicable, *e.g. banded*: how to decide which of two banded images is "more banded" than the other? Possible answers are: density of stripes, number of stripes, width of stripes, but the answer is not easy to define.

The result of this selection process is a dataset of 120 *key representative images* for each of the 47 attributes. The annotation pipeline could be further improved, for extending the dataset. Already having an initial set of images, we could use the scores from the classifiers learnt on the existing images to decide for which of the new images to collect annotations.

3.1.4 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 another 46 attributes for each of the 5,640 texture images is fairly expensive. To reduce this cost we propose to exploit the correlation and sparsity of the attribute occurrences (Figure 3.5). Twelve key images for each attribute q, are annotated exhaustively

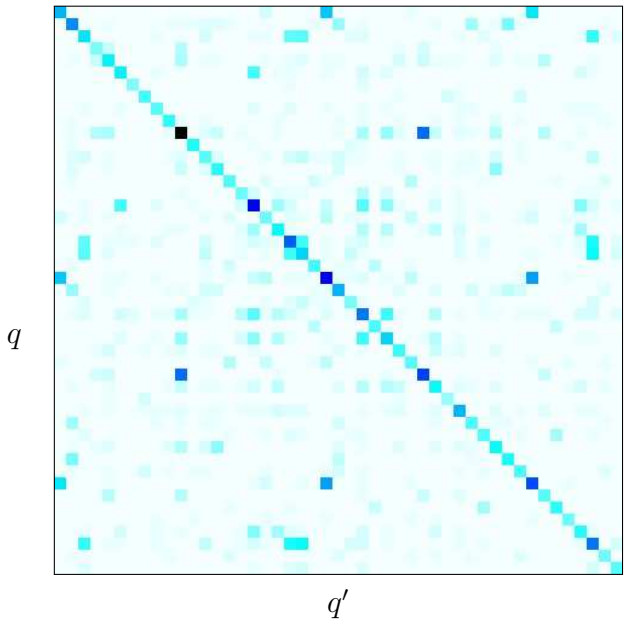


Figure 3.5: **Co-occurrence of attributes.** The matrix shows the joint probability $p(q, q')$ of two attributes occurring together (rows and columns are sorted alphabetically, as in the previous figure).

with all 47 attributes, 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 – in practice 4 or 5 – are collected, assuming that the rest of the attributes would not apply. This procedure occasionally misses attribute annotations; Figure 3.4 evaluates attribute recall by 12-fold cross-validation on the 12 exhaustive annotations for a fixed budget of collecting 10 annotations per image (instead of 47).

A further refinement is to suggest which attributes q' to annotate not just based on q, but also based on the appearance of an image ℓ_i. This was done by using the attribute classifier learned in Section 4.2, and incorporating the score of attributes in the decision process; after Platt's calibration [Platt, 2000], the classifier score $c_{q'}(\ell_i) \in \mathbb{R}$ is transformed in a probability $p(q'|\ell_i) = \sigma(c_{q'}(\ell))$ where $\sigma(z) = 1/(1 + e^{Az+B})$ is the sigmoid function, with parameters A and B learnt on an held-out test

set. To reflect the probability $p(q'|q)$ instead, a score is computed 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 to annotate for each image. The score could be seen as a probability, by using a normalization constant. In this score, we account for the classification score for attribute q' for the given image ℓ_i, as well as the co-occurrence probability $p(q'|q)$, and inversely proportional to the probability of the attribute q' not occurring, given the key attribute q. Although the value of $\frac{1-p_0}{p_0}$ is a constant, equal to 46, and it only scales the proportionality constant, we included it to reflect the prior probability of co-occurrence, based on the assumption that textures are uniformly distributed in the dataset, by design.

As shown in Figure 3.4, for a fixed annotation budged this method increases attribute recall. Overall, with roughly 10 annotations per images 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.

3.1.5 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 Chapter 5.

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 *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 normalised 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

$$\mathrm{acc}(\hat{c}) = \frac{1}{C} \sum_{\bar{c}=1}^{C} \frac{|\{\mathbf{x} : c(\mathbf{x}) = \bar{c} \wedge \hat{c}(\mathbf{x}) = \bar{c}\}|}{|\{\mathbf{x} : c(\mathbf{x}) = \bar{c}\}|}. \tag{3.1}$$

mAP is defined as for the PASCAL VOC 2008 benchmark onward Everingham et al., 2008.

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.

3.2 Textures and Materials in Clutter

The previous section introduced our first contribution in the study of the texture patterns, namely the study of describable texture attributes. This section looks instead at our second contribution: 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 sample fills the entire image, which is not useful in 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.

Masks are provided in FMD, but since most of the images are covered entirely by one material, in practice the utility of the masks is small (our own methods don't use masks). In addition, FMD is a relatively small dataset (1000 images, 10 classes). We remove this last-standing limitation of isolating the texture and materials in

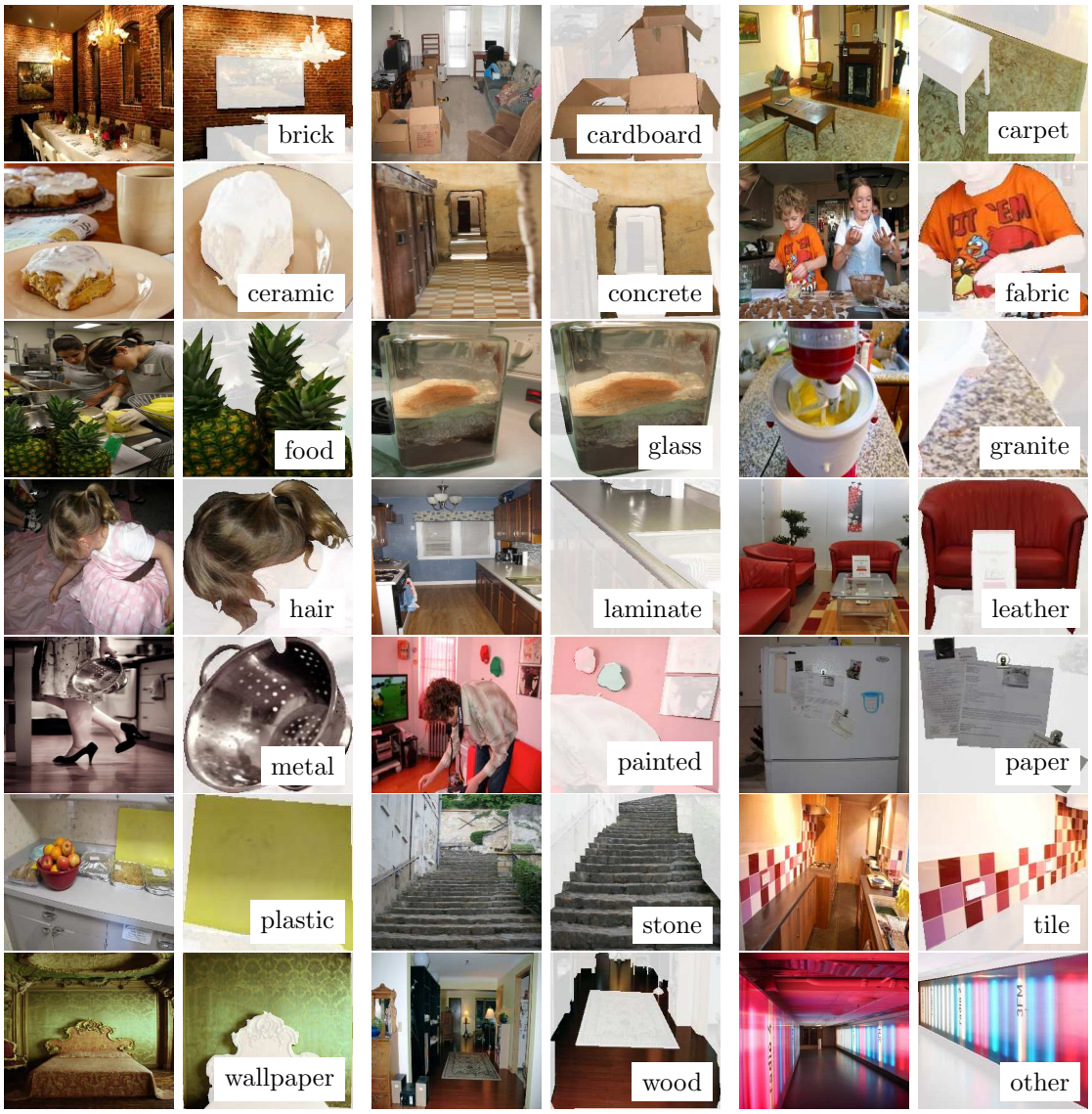


Figure 3.6: Example material segments from the Open Surfaces dataset. For each category, we show two images, the left one representing the source image, from which the segments shown on the right were extracted. Note the variations in scale, illumination, viewing angle, even within the same material segment.

images, by experimenting with a *large* dataset of textures collected in the *wild* and in *cluttered conditions*.

Rather than collecting a new image dataset, the 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 annotation. 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 22 different classes[1], from which we show some examples in Figure 3.6. Images are split evenly into training, validation, and test subsets with 3,474 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 crowd-sourced 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.

Finally, in order to study perceptual properties as well as materials, the OS dataset was augmented with some of the describable attributes of Section 3.1. 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.[2] The attributes were manually labelled in the 53,915 segments retained for materials. We refer to this data as OSA.

3.2.1 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/dtd/`.

[1]The classes and corresponding number of example segments are: brick (610), cardboard (423), carpet/rug (1,975), ceramic (1,643), concrete (567), fabric/cloth (7,484), food (1,461), glass (4,571), granite/marble (1,596), hair (443), other (2,035), laminate (510), leather (957), metal (4,941), painted (7,870), paper/tissue (1,226), plastic/clear (586), plastic/opaque (1,800), stone (417), tile (3,085), wallpaper (483), wood (9,232).

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

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 OSR+R is measured in term of classification accuracy and mAP, using the same definition (3.1) 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*, also instantiate for materials (OS) and describable texture attributes (OSA). Since not all image pixels are labelled in the ground truth, 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 (3.1) 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, per-pixel accuracy is modified such that a class prediction is considered correct if it belongs to any of the ground-truth pixel labels.

Chapter 4

Texture Representations

This chapter discusses texture recognition methods, setting up a framework for their systematic evaluation and comparison. Texture recognition starts from the problem of representing images of textures. 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. Classification 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, in this thesis we are particularly interested in the class of *texture representations* pioneered by the work of Malik and Perona, 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 modelling 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 beyond textures, or inspiring methods applied on other tasks. For example, until recently most 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. One of the contributions of this work is to show that these broadly-applicable texture representation variants are in fact optimal for a large number of texture-specific problems as well, often surpassing dedicated representations (Section 5.1.3).

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; Girshick et al., 2014] can be interpreted as extracting local image descriptors (performed by 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 mechanism 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 down in 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 honour of the pioneering work of 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*.

Section 4.1 and Section 4.2 discuss filter banks and pooling encoders in detail, while in Section 4.2.4 we are proposing a hybrid architecture, that combines the CNN features as local descriptors, but uses a standard encoder for pooling, in our case, Fisher Vector, but could be any of the encoders described in Section 4.2.

4.1 Local Image Descriptors

There is a vast choice of local image descriptors in texture representations. Traditionally, these features were handcrafted, but with the latest generation of deep learning methods it is now customary to learn them from data. These two families of local features are discussed in Section 4.1.1 and Section 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 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 quantization (Section 4.2.1), were used as the computational basis of the "textons" proposed by Julesz [Julesz and Bergen, Jul-Aug 1983]. The **MR8** filter bank [Varma and Zisserman, 2003; 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 [Varma and Zisserman, 2003]. The **VZ** descriptors are in fact small image patches which, remarkably, were shown to outperform LM and MR8 on typical texture benchmarks such as CuRET that were popular at that time. However, as will be demonstrated in the experiments, trivial local descriptors are not competitive in harder tasks.

Another early local image descriptor is the **Local Binary Patterns** (LBP) [Ojala et al., 1996; 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 at 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 neighbours $\mathbf{p}_j$ (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 pat-*

terns [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 **SIFT** [Lowe, 1999]. SIFT is the histogram of the occurrences of image gradients quantized with respect to their location within a patch and their orientation. While SIFT was originally introduced to match object instances, it was later applied to an impressive diversity of different tasks, from object categorization to semantic segmentation and face recognition.

4.1.2 Learned Local Features

Hand-crafted 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; Section 4.2.1) or whether they reflect it (order-sensitive pooling; Section 4.2.2). A detail of practical importance, furthermore, is the type of post-processing applied to the pooled vector (Section 4.2.3).

4.2.1 Orderless Pooling Encoders

An *orderless pooling encoder* ϕ_e maps a collection $\mathcal{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_n), \mathbf{f}_i \in \mathbb{R}^D$ 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 permutation of the input $\mathcal{F}$. [1] Furthermore, the encoder can be applied to any number of features; for example, the encoder can be applied to the subset $\mathcal{F}' \subset \mathcal{F}$ of local descriptors that apply to 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 $C = \begin{bmatrix} \mathbf{c}_1 & \ldots & \mathbf{c}_K \end{bmatrix} \in \mathbb{R}^{D \times K}$ of K

[1] Note that $\mathcal{F}$ cannot be represented as encoders are generally sensitive to feature repetitions. Instead, ϕ_e is invariant to permutations of the sequence $\mathcal{F}$.

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 by Sivic and Zisserman, 2003 and Csurka et al., 2004 respectively, for object instance and category understanding, inspired from text retrieval. Earlier work of Leung and Malik, 2001 also used histograms to characterise the distribution of textons, defined as configuration of local filter responses, at each pixel. 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: $[\eta_{\mathrm{KC}}(\mathbf{f}_i)]_j \propto \exp\left(-\lambda\|\mathbf{f}_i - \mathbf{c}_j\|^2\right)$, where λ is a parameter controlling the locality of the assignment. The descriptor code $\eta_{\mathrm{KC}}(\mathbf{f}_i)$ is L^1 normalised before aggregation, such that $\|\eta_{\mathrm{KC}}(\mathbf{f}_i)\|_1 = 1$.

Sparse Coding [Olshausen and Field, 1997] was introduced first in neuroscience, in the context of explaining the functionality of simple cells in visual cortex of mammals. Later on, several algorithms for dictionary learning were developed [Mairal et al., 2008; Yang et al., 2010], with various applications, like image reconstruction.

Locality constrained Linear Coding (LLC) [Wang et al., 2010] is one of several methods which used *sparse coding* to define the local descriptor encoder [Zhou et al., 2010; Liu et al., 2011]. It extends this idea of kernel codebook encoding, by making the assignments reconstructive, local, and sparse: the descriptor encoder $\eta_{\mathrm{LLC}}(\mathbf{f}_i) \in \mathbb{R}^d_+$, $\|\eta_{\mathrm{LLC}}(\mathbf{f}_i)\|_1 = 1$, $\|\eta_{\mathrm{LLC}}(\mathbf{f}_i)\|_0 \leq r$ is computed such that $\mathbf{f}_i \approx C\eta_{\mathrm{LLC}}(\mathbf{f}_i)$ while allowing non-zero coefficients only to the $r \ll K$ visual words closest to $\mathbf{f}_i$.

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, and C denotes the nearest word in the codebook, obtained via K-means. 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; 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 descriptors, 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 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.* a 16×16 grid), 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 $\mathbf{y} = \bar{\phi}_e(\mathcal{F})$ 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 normalisation $\phi_e(\mathcal{F}) = \mathbf{y}/\|\mathbf{y}\|_2$. 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*[2] $\phi_{\text{Hell.}}(y) = \text{sign}(y)\sqrt{|y|}$.

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 normalising the kernel embedding $\phi_K(\mathbf{y})$ amounts to normalising 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 normalisation. VLAD has several standard variants that differ in the post-processing; here we use the one that L^2 normalises the individual VLAD subvectors (one for each visual word) before L^2 normalising the whole vector [Arandjelovic and Zisserman, 2012].

4.2.4 Hybrid representations - FV-CNN

We have seen in the previous sections a summary of the existing local descriptors and encodings, whose combinations were widely used in the literature. The pipeline

[2]This extension generalises to all homogeneous kernel, including for example χ^2 [Vedaldi and Zisserman, 2010].

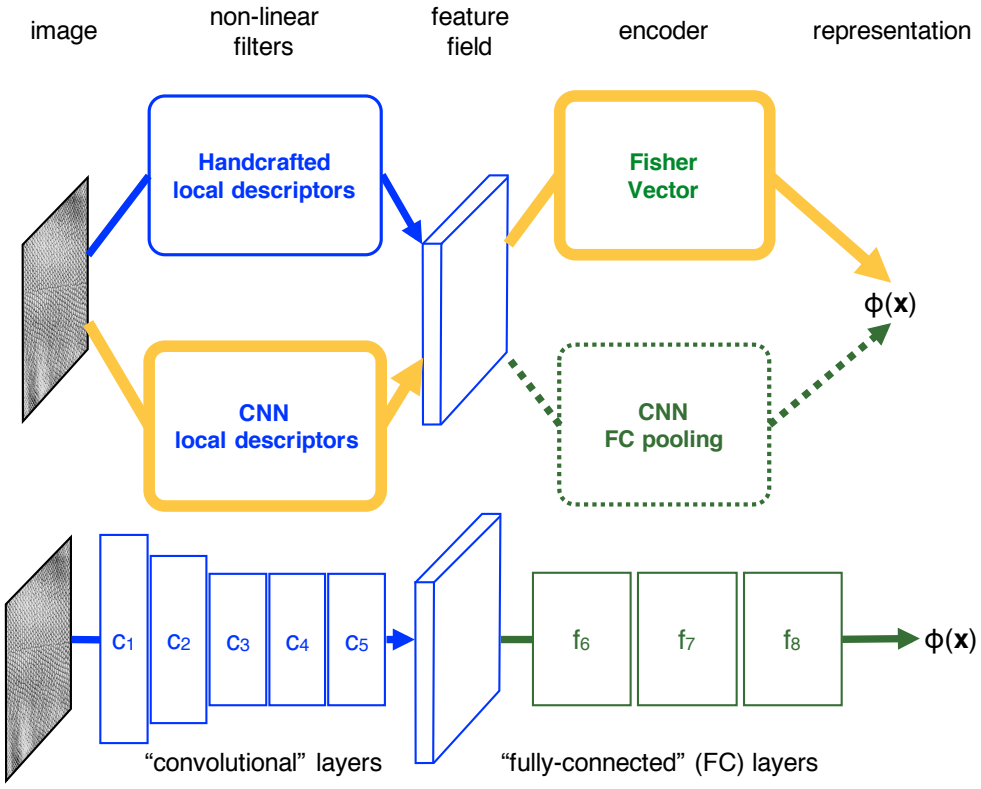


Figure 4.1: **FV-CNN, a hybrid image representation.** The similarity between CNN and DESCRIPTOR-ENCODER pipelines (top) enables obtaining hybrid archidectures, by interchangning the building blocks. Following the upper path, we obtain the standard Local descriptor-Pooling Encoder approach; the bottom part gives the CNN pipeline; we highlighted in yellow FV-CNN, proposed in this thesis, which explores the alternate path. (bottom) The CNN architecture proposed by Krizhevsky et al., 2012

is as follows: from an image, we extract local descriptors, using linear or non-linear filters. Then, we pool the feature field in an order-less manner, using a suitable encoder, like Bag of Visual Words, or Fisher Vector.

However, these days the most popular image representations are based on convolutional neural networks. We are interested to see how we could reuse elements of classical texture representations in the context of deep learning. We note that a CNN is also a sequence of operators, like in the standard pipeline. The first set of operators (or layers), called convolutional, are local and translation invariant; the

last operators, called instead fully connected, operate on the whole image. Therefore, it is natural to think of the convolutional layers as local image descriptors, similar to SIFT, and to see the fully connected layers as a pooling encoder. Although, mathematically, convolutional and fully connected layers are essentially the same, it is the information encoded by them, local or global, which makes our decomposition natural. It is also important to note that the fully connected layers provide order-sensitive statistics, which may be less useful for texture recognition, especially for repetitive patterns.

Given this insight, the two pipelines are similar, and their blocks can be interchanged. This enables following the standard, CNN or local descriptors and encoder pipelines, but also hybrid architectures, as shown in Figure 4.1. One such architecture is **FV-CNN**, which uses CNN local descriptors, but replaces the pooling from the fully connected layers, with a classical orderless encoder. As we will see in the next chapter (Section 5.1.4 and 5.1.5), using Fisher Vector as encoder, and features from the last convolutional layer as local descriptors, gives the best results across several benchmarks from a wide range of domains: texture, material, object recognition, scene understanding and even fine-grained categorization.

Besides the path explored in this thesis, there is one more possible way to interchange the blocks, that is, to use standard local features, such as SIFT, and fully connected layers of a CNN acting as a pooling encoder [Perronnin and Larlus, 2015].

Chapter 5

Experiments on Semantic Recognition

In the previous chapters, we introduced novel problems in texture understanding and reviewed a number of old and new texture representations. The goal of this chapter is to determine, through extensive experiments, what representations work best for which problem.

Representations are labelled as pairs ENCODER-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; Chatfield et al., 2014] implicitly use CNN-based descriptors and the FC pooler, and therefore are denoted here as FC-CNN.

The first set of experiments (Section 5.1) evaluates several local image descriptors and pooling encoder combinations on a small number of datasets in order to determine noteworthy representations. The second set of experiments (Section 5.2) evaluates the latter on a wide range of image recognition benchmarks in order to

establish their relative merits and breadth of applicability. We focus primarily on textures and materials (Section 5.2.1), but also evaluate the proposed method on coarse and fine-grained object categorization, semantic region recognition, and scene categorization (Section 5.2.2), and show the benefits of our method in domain transfer (Section 5.2.3).

5.1 Local Image Descriptors and Encoders Evaluation

This section compares different local image descriptors and pooling encoders (Section 5.1.1) on selected representative tasks in texture recognition, object recognition, and scene recognition (Section 5.1.2). In particular, Section 5.1.3 compares different local descriptors, Section 5.1.4 different pooling encoders, and Section 5.1.5 additional variants of the CNN-based descriptors.

5.1.1 General Experimental Setup

The experiments are centred 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 Caffe [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 64K 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 PCA-reducing FV to 4096 dimensions and observing only a marginal reduction in classification performance in the PASCAL VOC object recognition task 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 normalisation), C was simply set to the constant 1. Instead, it was found that recalibrating the SVM scores for each class improve classification accuracy (but not mAP). We recalibrated the scores 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 this thesis use the VLFeat library [Vedaldi and B. Fulkerson, 2008] for the computation of SIFT features and the pooling embeddings (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.

5.1.2 Datasets and Evaluation Measures

The evaluation is performed on a diversity of tasks: the new describable attribute and material recognition benchmarks in DTD and Open Surfaces, 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 (Section 3.1) contains 47 texture classes, one per visual attribute, containing 120 images each. Images are equally spilt into train, test and validation. The evaluation focuses on recognizing the "key attributes", as defined in Section 3.1.5 and reports accuracy averaged over the 10 default splits provided with the datasets. As shown later, in Table 5.4, recognizing the "joint attributes" gives similar results to recognizing the "key attributes".

Open Surfaces [Bell et al., 2013] is used in the setup described in Section 3.2.1 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 are reporting mean average precision, as this is a multi-label dataset. **FMD** [Sharan et al., 2009] consists of 1000 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** [Mallikarjuna et al., 2005] contains 4752 images, grouped into 11 material categories. For each material category, images of four samples were captured under various conditions, resulting 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 6700 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 provide 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.[1]

5.1.3 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-

[1]The definition of AP was changed in later versions of the benchmark.

Local descriptor	Descriptor dimensionality	Kernel			
		Linear	Hellinger	add-χ^2	exp-χ^2
MR8	8	20.8 ± 0.9	26.2 ± 0.8	29.7 ± 0.9	34.3 ± 1.1
LM	49	26.7 ± 0.9	34.8 ± 1.2	39.5 ± 1.4	44.0 ± 1.4
$\text{Patch}_{3\times3}$	9	15.9 ± 0.5	24.4 ± 0.7	27.8 ± 0.8	30.9 ± 0.7
$\text{Patch}_{7\times7}$	49	20.7 ± 0.8	30.6 ± 1.0	34.8 ± 1.0	37.9 ± 0.9
LBP^u	58[1]	8.5 ± 0.4	9.3 ± 0.5	12.5 ± 0.4	19.4 ± 0.7
LBP-VQ	58[1]	26.2 ± 0.8	28.8 ± 0.9	32.7 ± 1.0	36.1 ± 1.3
SIFT	80[2]	**45.2 ± 1.0**	**49.1 ± 1.1**	**50.9 ± 1.0**	**52.3 ± 1.2**
Conv VGG-M	512	**55.9 ± 1.3**	**61.7 ± 0.9**	**61.9 ± 1.0**	**61.2 ± 1.0**
Conv VGG-VD	512	**64.1 ± 1.3**	**68.8 ± 1.3**	**69.0 ± 0.9**	**68.8 ± 0.9**

Table 5.1: Comparison of local features and kernels on the DTD data. 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. [1] VLFeat implementation is 58D, [2] SIFT is reduced from 128D to 80D via PCA.

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 task of recognizing describable texture attributes in DTD (Section 3.1 using the BoVW encoder. As a byproduct, the experiments determine the relative difficulty of recognizing the different 47 perceptual attributes in DTD.

Experimental setup. The following local image descriptors are compared: the linear filter banks of *Leung and Malik* (LM) (48D descriptors) [Leung and Malik, 2001] and *MR8* (8D descriptors) [Geusebroek et al., 2003; Varma and Zisserman, 2005], 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), *DSFIT* (128D), and CNN features extracted from *VGG-M* and *VGG-VD* (512D).

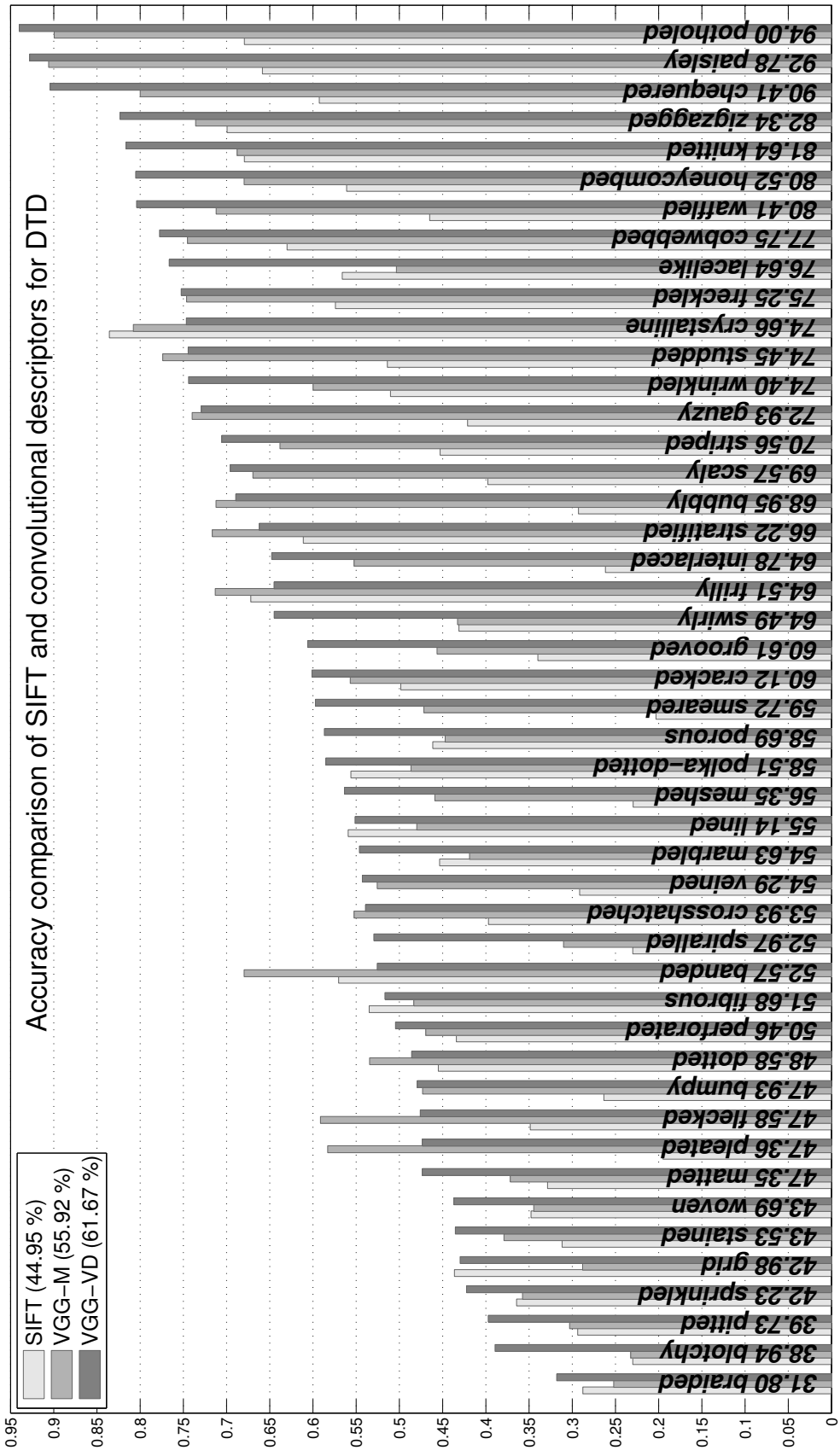


Figure 5.1: Per class classification accuracy in the DTD data comparing three local image descriptors: SIFT, VGG-M, and VGG-VD. For all three local descriptors, BoVW with 4096 visual words was used. Classes are sorted by increasing BoVW-CNN-VD accuracy (this number is reported along each bar).

After the BoVW representation is extracted, it is used to train a 1-vs-all SVM using the different kernels discussed in Section 4.2.3: linear, Hellinger, additive-χ^2, and exponential-χ^2. Kernels are normalised 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 BoVW vectors are L^1 normalised. An important parameter in BoVW is the number of visual words selected. K was varied in the range 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.

Analysis. Table 5.1 reports the classification accuracy for 47 1-vs-all SVM attribute classifiers, computed as in Section 3.1. 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 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 5.1 reports the classification accuracy for each attribute in DTD for the BoVW-SIFT, -VGG-M, and -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.

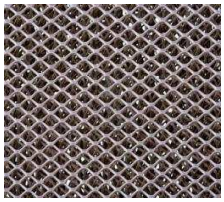
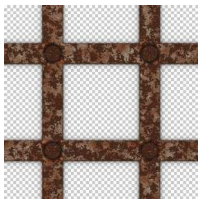
Image	Feature		
	SIFT	VGG-M	VGG-VD
	**grid**, lacelike, *meshed*, waffled, fibrous, porous, flecked, zigzagged, lined, bumpy	fibrous, **grid**, flecked, *meshed*, marbled, blotchy, bubbly, porous, matted, pitted,	*meshed*, fibrous, flecked, marbled, pitted, grooved, woven, dotted, polka-dotted, lined
	grid, interlaced, chequered, woven, marbled, flecked, polka-dotted, porous, swirly, grooved	chequered, blotchy, woven, sprinkled, knitted, lined, wrinkled, polka-dotted, scaly, honeycombed	blotchy, stained, perforated, porous, pitted, sprinkled, crosshatched, honeycombed, chequered, polka-dotted
	**fibrous**, *matted*, stratified, porous, paisley, frilly, *gauzy*, freckled, smeared, interlaced	marbled, *gauzy*, stained, crystalline, **fibrous**, scaly, paisley, bumpy, bubbly, pitted	marbled, smeared, *matted*, **fibrous**, braided, swirly, cobwebbed, crystalline, wrinkled, honeycombed
	lined, grooved, flecked, stratified, pleated, gauzy, dotted, knitted, marbled, polka-dotted	lined, marbled, grooved, flecked, bumpy, blotchy, knitted, crosshatched, matted, braided	grooved, lined, pleated, crosshatched, woven, pitted, sprinkled, bumpy, matted, marbled
	crystalline, frilly, gauzy, bumpy, cracked, studded, honeycombed, scaly, braided, veined	crystalline, bumpy, potholed, studded, braided, crosshatched, wrinkled, veined, cracked, scaly	wrinkled, crystalline, stained, bumpy, smeared, crosshatched, studded, gauzy, potholed, braided
	banded, porous, pleated, flecked, cobwebbed, marbled, interlaced, perforated, spiralled, knitted	pleated, banded, lined, interlaced, braided, striped, meshed, flecked, matted, knitted	lined, banded, pitted, crosshatched, marbled, matted, polka-dotted, woven, braided, meshed

Table 5.2: **SIFT vs CNN local descriptors**. Examples of images for which SIFT outperforms CNN local features. In bold, we marked the ground-truth label, and in italic, reasonable predictions.

While on average, convolutional features significantly outperform SIFT, and the CNN features from a deeper model give a further 10% improvement, there are some cases in which, surprisingly, SIFT is performing better: *grid, lined, polka-dotted, crystalline*, and some cases in which the features from VGG-M are outperforming VGG-VD features: *banded, pleated, polka-dotted, studded*. It is important to note that the number of images in the test set is relatively small (40 per class), and a 15% difference in per-class accuracy, as in the case of grid, means 6 images which were not classified correctly using CNN features, and were correctly classified using SIFT. Looking closely at these images, we note that the CNN features give reasonable top guesses – *e.g.* suggesting terms like *meshed* for an image labelled as grid, or *chequered, blotchy, stained* in the case of the second image in Table 5.2, reflecting details in the image: chequered background, stained metal forming the grid. The top guesses from convolutional features are either classes with similar aspect (grid vs meshed, lined vs banded), or describe some details in the appearance of the image. Another interesting example is the *crystalline* image, recognized as *wrinkled* – similar in appearance and colour with aluminium foil, present in the examples for wrinkled class.

Although for certain classes there are a few images which are classified correctly for SIFT features, for the majority of classes, CNN features perform better, and the gap is wider (*e.g.* for *pleated*, VGG-M features are better by 35% absolute accuracy than the SIFT counterpart).

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.

5.1.4 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 (Section 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 of the FC layers of a network.

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's 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 much smaller codebook as these representation multiply the dimensionality of the descriptors (Section5.1.1). 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 analysed in Section 5.1.5.

Before pooling local descriptor 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. However, PCA is not applied to the convolutional features as it was shown to deteriorate performance in this case. The improved version of the FV is used in all the experiments in this chapter, and, similarly, for VLAD, we applied signed square root to the resulting encoding, which is then normalised component-wise (Section 4.2.3).

Dataset	meas. (%)	SIFT				VGG-M				VGG-M
		BoVW	LLC	VLAD	IFV	BoVW	LLC	VLAD	IFV	FC
DTD	acc	$49.0_{\pm 0.8}$	$48.2_{\pm 1.4}$	$54.3_{\pm 0.8}$	$58.6_{\pm 1.2}$	$61.2_{\pm 1.3}$	$64.0_{\pm 1.3}$	$\mathbf{67.6_{\pm 0.7}}$	$\mathbf{66.8_{\pm 1.5}}$	$58.7_{\pm 0.9}$
OS+R	acc	30.0	30.8	32.5	39.8	41.3	45.3	49.7	**52.5**	41.3
KTH-T2b	acc	$57.6_{\pm 1.5}$	$56.8_{\pm 2.0}$	$64.3_{\pm 1.3}$	$70.2_{\pm 1.6}$	$73.6_{\pm 2.8}$	$74.0_{\pm 3.3}$	$72.2_{\pm 4.7}$	$\mathbf{73.3_{\pm 4.8}}$	$71.0_{\pm 2.3}$
FMD	acc	$50.5_{\pm 1.7}$	$48.4_{\pm 2.2}$	$54.0_{\pm 1.3}$	$59.7_{\pm 1.6}$	$67.9_{\pm 2.2}$	$71.7_{\pm 2.1}$	$\mathbf{74.2_{\pm 2.0}}$	$\mathbf{73.5_{\pm 2.0}}$	$70.3_{\pm 1.8}$
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

Table 5.3: Pooling encoder comparisons. 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.

Analysis. Results are reported in Table 5.3. In term of orderless encoders, BoVW and LLC result in similar performance for SIFT, while the difference is slightly larger and in favour of LLC for CNN features. Note that BoVW is used with Hellinger kernel, which contributes to reducing the gap between BoVW and LLC. IFV and VLAD significantly outperform BoVW and LLC in almost all tasks; FV is definitely better than VLAD 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 be confirmed in other experiments such as Table 5.4.

Next, we compare using CNN features with an orderless encoder (FV-CNN) as opposed to the standard FC layer (FC-CNN). As seen in Table 5.3 and Table 5.4, 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 is easily explained by the fact that, differently from Chatfield et al., 2011, these experiments do not use SPP and image augmentation.

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.

5.1.5 CNN Descriptor Variants Comparison

This section conducts additional experiments on CNN local descriptors to find the best variants – from comparing network architectures, to varying depths at which features are extracted, and encoder parameters.

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

Analysis. Results are presented in detail in Table 5.4, with an emphasis on texture and material datasets (a, b and c), but conclusions are similar for the other datasets (d). 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 is a 7.1% improvement, while going from FC-VGG-M to FV-VGG-VD is a 11.3% improvement). Combining FV-CNN and FC-CNN (by concatenating 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 analyse in detail the effect of depth on the convolutional features. Figure 5.3 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 centres K is set such that the overall dimensionality of the descriptor $2KD_k$ is constant.

	dataset	meas. (%)	IFV	AlexNet			VGG-M			VGG-VD			FV-SIFT	SoA
				FC	FV	FC+FV	FC	FV	FC+FV	FC	FV	FC+FV	FC+FV-VD	
(a)	CUReT	acc	$99.0_{\pm 0.2}$	$94.4_{\pm 0.4}$	$98.5_{\pm 0.2}$	$99.0_{\pm 0.2}$	$94.2_{\pm 0.3}$	$98.7_{\pm 0.2}$	$99.1_{\pm 0.2}$	$94.5_{\pm 0.4}$	$99.0_{\pm 0.2}$	$99.2_{\pm 0.2}$	$\mathbf{99.7}_{\pm 0.1}$	$99.8_{\pm 0.1}$ [1]
	UMD	acc	$99.1_{\pm 0.5}$	$95.9_{\pm 0.9}$	$\mathbf{99.7}_{\pm 0.2}$	$\mathbf{99.7}_{\pm 0.3}$	$97.2_{\pm 0.9}$	$\mathbf{99.9}_{\pm 0.1}$	$\mathbf{99.8}_{\pm 0.2}$	$97.7_{\pm 0.7}$	$\mathbf{99.9}_{\pm 0.1}$	$\mathbf{99.9}_{\pm 0.1}$	$\mathbf{99.9}_{\pm 0.1}$	$99.7_{\pm 0.3}$ [1]
	UIUC	acc	$96.6_{\pm 0.8}$	$91.1_{\pm 1.7}$	$99.2_{\pm 0.4}$	$99.3_{\pm 0.4}$	$94.5_{\pm 1.4}$	$\mathbf{99.6}_{\pm 0.4}$	$99.6_{\pm 0.3}$	$97.0_{\pm 0.7}$	$\mathbf{99.9}_{\pm 0.1}$	$\mathbf{99.9}_{\pm 0.1}$	$\mathbf{99.9}_{\pm 0.1}$	$99.4_{\pm 0.4}$ [1]
	KT	acc	$\mathbf{99.5}_{\pm 0.5}$	$95.5_{\pm 1.3}$	$\mathbf{99.6}_{\pm 0.4}$	$\mathbf{99.8}_{\pm 0.2}$	$96.1_{\pm 0.9}$	$\mathbf{99.8}_{\pm 0.2}$	$\mathbf{99.9}_{\pm 0.1}$	$97.9_{\pm 0.9}$	$\mathbf{99.8}_{\pm 0.2}$	$\mathbf{99.9}_{\pm 0.1}$	**100**	$99.4_{\pm 0.4}$ [1]
	ALOT	acc	$94.6_{\pm 0.3}$	$86.0_{\pm 0.4}$	$\mathbf{96.7}_{\pm 0.3}$	$\mathbf{97.8}_{\pm 0.2}$	$88.7_{\pm 0.5}$	$\mathbf{97.8}_{\pm 0.2}$	$\mathbf{98.4}_{\pm 0.1}$	$90.6_{\pm 0.4}$	$\mathbf{98.5}_{\pm 0.1}$	$\mathbf{99.0}_{\pm 0.1}$	$99.3_{\pm 0.1}$	$95.9_{\pm 0.5}$ [2]
(b)	KTH-T2b	acc	$70.8_{\pm 2.7}$	$71.5_{\pm 1.3}$	$69.7_{\pm 3.2}$	$72.1_{\pm 2.8}$	$71_{\pm 2.3}$	$73.3_{\pm 4.7}$	$73.9_{\pm 4.9}$	$75.4_{\pm 1.5}$	$\mathbf{81.8}_{\pm 2.5}$	$\mathbf{81.1}_{\pm 2.4}$	$\mathbf{81.5}_{\pm 2.0}$	$76.0_{\pm 2.9}$ [2]
	FMD	acc	$59.8_{\pm 1.6}$	$64.8_{\pm 1.8}$	$67.7_{\pm 1.5}$	$71.4_{\pm 1.7}$	$70.3_{\pm 1.8}$	$73.5_{\pm 2.0}$	$76.6_{\pm 1.9}$	$77.4_{\pm 1.8}$	$79.8_{\pm 1.8}$	$\mathbf{82.4}_{\pm 1.5}$	$\mathbf{82.2}_{\pm 1.4}$	$57.7_{\pm 1.7}$ [3]
	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	$58.6_{\pm 1.2}$	$55.1_{\pm 0.6}$	$62.9_{\pm 1.4}$	$66.5_{\pm 1.1}$	$58.8_{\pm 0.8}$	$\mathbf{66.8}_{\pm 1.6}$	$\mathbf{69.8}_{\pm 1.1}$	$62.9_{\pm 0.8}$	$\mathbf{72.3}_{\pm 1.0}$	$\mathbf{74.7}_{\pm 1.0}$	$\mathbf{75.5}_{\pm 0.8}$	–
	DTD	mAP	$61.3_{\pm 1.1}$	$57.7_{\pm 0.9}$	$66.5_{\pm 1.4}$	$70.5_{\pm 1.2}$	$62.1_{\pm 0.9}$	$\mathbf{70.8}_{\pm 1.2}$	$\mathbf{74.2}_{\pm 1.1}$	$67.0_{\pm 1.1}$	$\mathbf{76.7}_{\pm 0.8}$	$\mathbf{79.1}_{\pm 0.8}$	$\mathbf{80.4}_{\pm 0.9}$	–
	DTD-J	mAP	$59.6_{\pm 0.6}$	$58.4_{\pm 0.7}$	$65.0_{\pm 0.9}$	$68.3_{\pm 0.9}$	$62.8_{\pm 0.7}$	$\mathbf{69.8}_{\pm 0.9}$	$\mathbf{72.9}_{\pm 0.9}$	$67.3_{\pm 0.9}$	$\mathbf{75.8}_{\pm 0.6}$	$\mathbf{77.5}_{\pm 0.8}$	$\mathbf{78.9}_{\pm 0.7}$	–
	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 [4]
	MIT Indoor	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
	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 (62.8^{*}) [6]
	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 [6]

Table 5.4: State-of-the art texture descriptors. 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. [1] Sifre and Mallat, 2013, [2] Sulc and Matas, 2014, [3] Qi et al., 2014, Sharan et al., 2013 – using a combination of features, [4] Wei et al., 2014, [5] Zhou et al., 2014, [6] Zhang et al., 2014

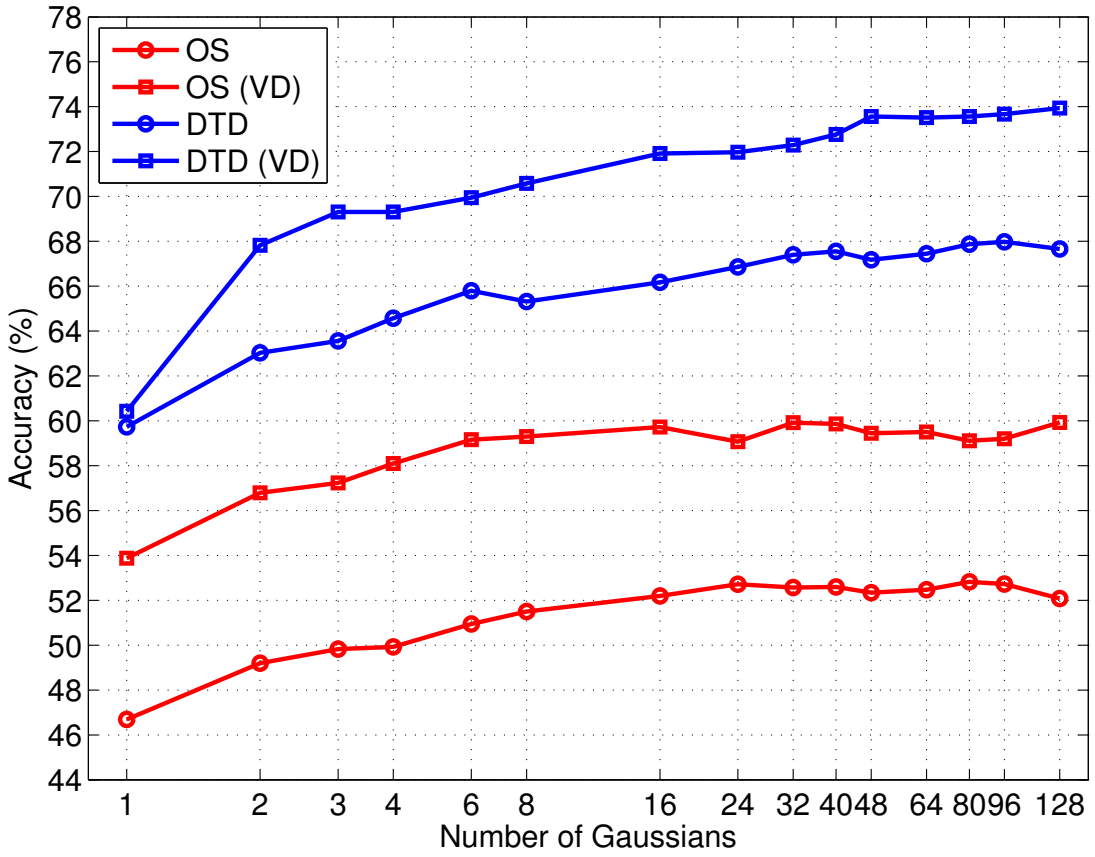


Figure 5.2: **Effect of the number of Gaussian components in the FV encoder.** The figure shows the performance of the FV-VGG-M and FV-VGG-VD representations on the OS and DTD datasets when the number of Gaussians components in the GMM is varied from 1 to 128.in the GMM, from 1 to 128. Note that abscissa is scaled logarithmically.

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.

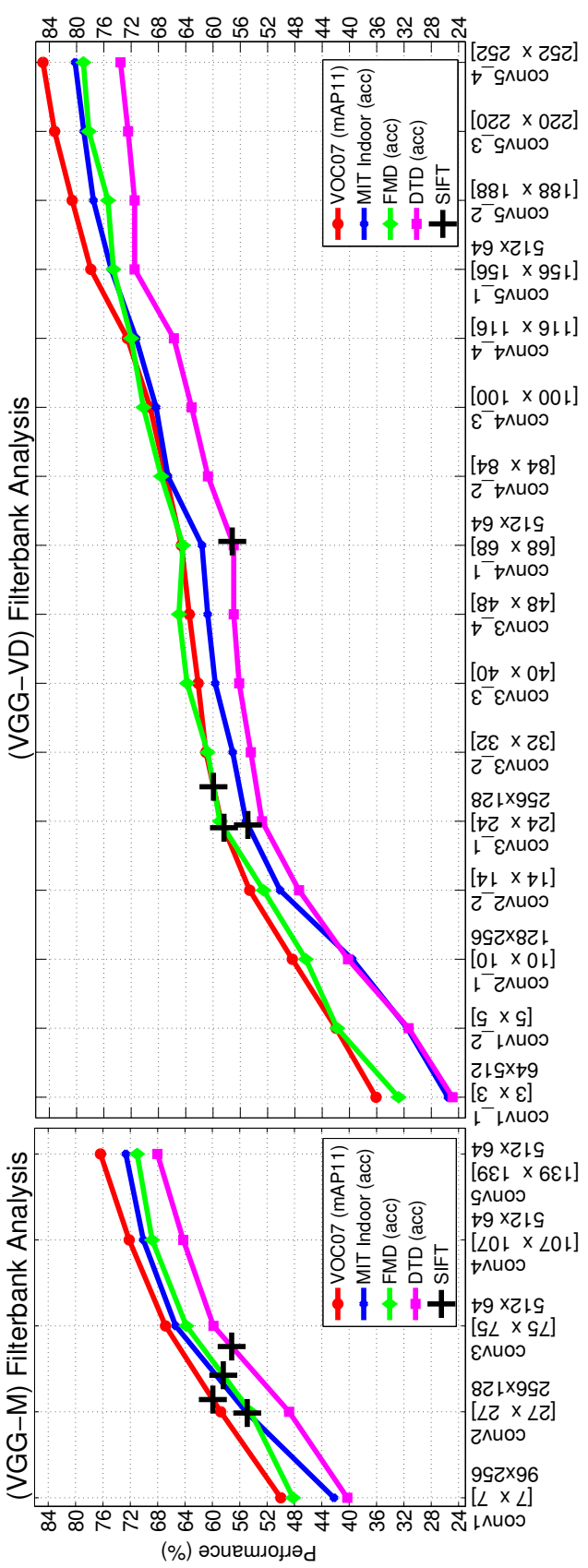


Figure 5.3: **Effect of the depth on CNN features.** The figure reports the performance of VGG-M (left) and VGG-VD (right) local image descriptors pooled with the FV encoder. For each layer the figures shows the size of the receptive field of the local descriptors (denoted $[N \times N]$]), as well as 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. For VGG-VD, we display the dimension of local descriptors and the number of visual words in the FV representation only for the first layer of the convolutional block.

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 Figure 5.2. 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.

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 realised in the very last few layers.

5.2 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 (Section 5.2.1) to object and scene recognition (Section 5.2.3). It also emphasises 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 Section 6.2.

5.2.1 Texture Recognition

We are grouping the experiments on textures in this section, by the conditions in which the images were collected, and by the area covered by the texture, as follows:

recognition in *controlled conditions*, where the main variables are viewpoint and illumination, recognition *in the wild*, characterised by larger intra-class variations, and recognition *in the wild and clutter*, where textures are a small portion of a larger scene.

Datasets and evaluation measures. In addition to the texture datasets evaluated in Section 5.1, 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 it 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. 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.

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

Texture recognition in controlled conditions. The results for texture representations on datasets which are collected under controlled conditions are shown in section (a) of Table 5.4.

For instance recognition, CUReT, UIMD, UIUC are saturated by modern techniques such as [Sifre and Mallat, 2013; Sharma et al., 2012; 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 loses about (3%) accuracy compared to FV-CNN.

In material recognition, KTH-T2b and ALOT are more challenging compared to the other benchmarks, KTH-T2b exposing intra-class variation, as opposed to classifying texture instances, and ALOT having a large number of classes – 250. First, there is a significant gap 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([Timofte and Van Gool, 2012]) 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 behind FV-VGG-VD by +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.

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 (∼3%) on FMD and substantially better in DTD (∼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%).

Concatenating FC-CNN and FV-CNN improves performance by about 3% across the board, showing that these features are somewhat complementary, 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 our previous finding that SIFT was fairly complementary to FC-CNN on a variant of AlexNet in Cimpoi et al., 2014.

Texture recognition in clutter. In what follows, we evaluate 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 Section 3.2.1. Recall that the +R suffix indicates that, while textures are imaged in clutter, the classifier is

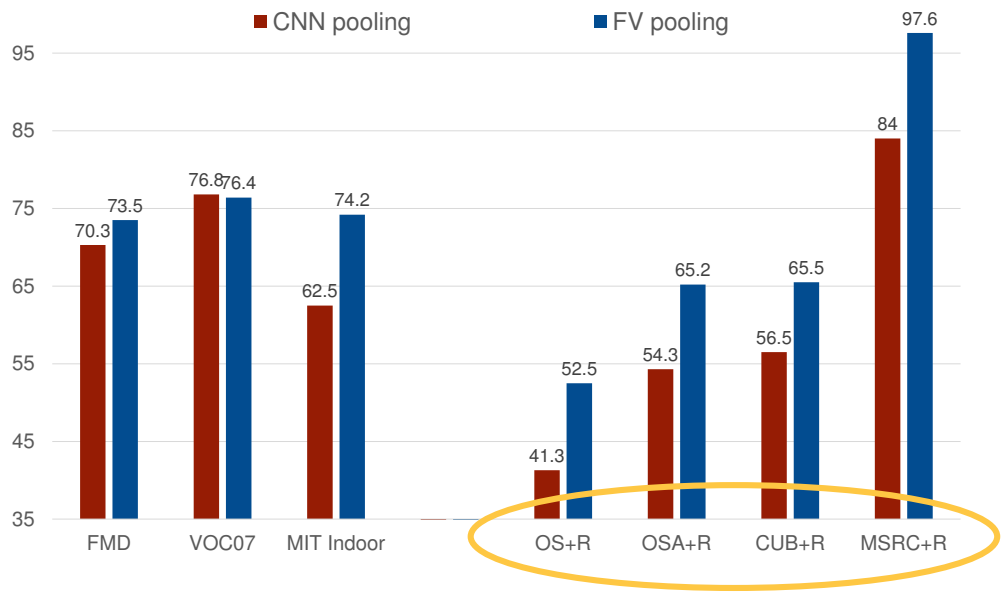


Figure 5.4: **Image recognition vs region recognition**. The advantage of using FV pooling instead CNN pooling becomes clear when the task is to classify regions of variable shapes and sizes. On the left, we show benchmarks for which the task is to classify the whole image, and on the right we show results for benchmarks in which the task is to classify image regions. We used the "+R" mark to indicate that regions to be classified are given. Note that on the right, the gap between FC-pooling and FV-pooling is bigger than on the left.

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 Section 6.2.

Results are reported in Table 5.4 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%). The advantage of orderless pooling of CNN local descriptors over FC pooling, when region of different sizes and shapes must be evaluated becomes is clear, when we compare the gap with results on benchmarks that require image classification as a

whole. These are visualized in Figure 5.4 – the gap is much smaller on the left, for the whole image recognition tasks, and visibly larger on the right, where we evaluate classification of regions of variable shape and size.

The significant computational advantage (evaluated further in Section 5.2.3) becomes obvious when several regions must be classified, as FV-pooled CNN features need not to be recomputed for each region, being shared across regions. Results on OSA+R are entirely analogous.

5.2.2 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.

Datasets and evaluation measures. In addition to the datasets seen before, here we experiment with fine grained recognition in the CUB data [Wah et al., 2011]. 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 extracted at multiple scales, but 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 Section 3.2.1.

However, this measure is further modified such that it is *not* normalised per class:

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

Analysis. Results are reported in Table 5.4 section d. On PASCAL VOC, MIT Indoor, CUB, and CUB+R the relative performance of the different descriptors is similar to what 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 simpler and more general 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 ($\sim$ 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 descriptor* 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.

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.

5.2.3 Domain transfer

This section investigates in more detail the problem of domain transfer in CNN-based features. CNNs require large amounts of data for training, which, for many target domains, is not available. To compensate the lack of training data, a model is pretrained on a large dataset, like Imagenet ILSVCR, and the resulting net – after cutting the last layer, is used to extract features. These features are then used to predict classes on the target dataset. We will call this approach *late transfer*, because we cut deep down in the network, in the area of the fully connected layers.

Using Fisher Vector for pooling local descriptors allows us to cut after the convolutional layers, and use their outputs as local features, then, use the resulting feature vectors to predict classes on a target domain. We will refer to this approach as *early transfer*, because we cut the network earlier, in the region of convolutional layers. Figure 5.5 illustrates the difference between the two types of transfer – the red arrow denotes the late transfer, and the blue arrow marks the early transfer.

To investigate the effect of the source domain on performance, we consider, in addition to the networks used so far, which are pretrained on ImageNet data, a new network, having the same architecture as AlexNet, but trained on PLACES dataset [Zhou et al., 2014], which consists of about 2.5 million labelled images of scenes. Zhou et al., 2014 showed that, applied to the task of scene recognition in MIT Indoor, these features outperform AlexNet [Krizhevsky et al., 2012] – a fact explained by the similarity of domains. We repeat this experiment using FC- and FV-CNN descriptors on top of VGG-M, VGG-VDs.

Results are shown in Table 5.5. The FC-CNN performance is in line with those reported in Zhou et al., 2014 – in scene recognition with FC-CNN the same CNN

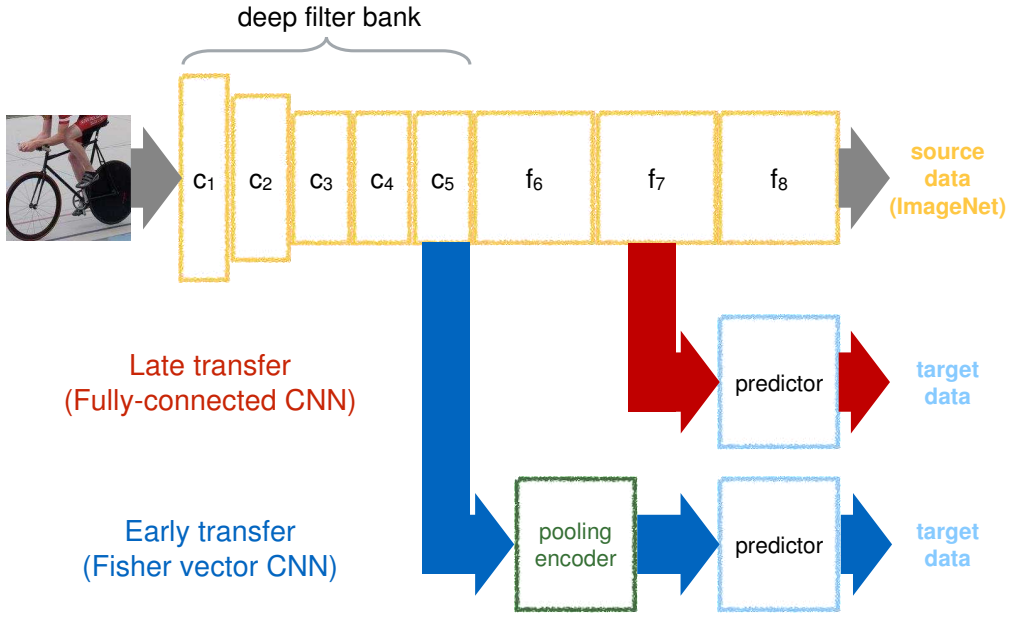


Figure 5.5: **Late vs. early transfer**. In red, we marked the standard, *late* transfer, that is, using fc_7 as features for images from the target domain. The blue arrow denotes the *early* transfer path, that is, cutting after a convolutional layer, and replacing the convolutional layers with a pooling encoder.

architecture performs better if trained on the Places dataset instead of the ImageNet data (58.6% vs 65.0% accuracy[2]). Nevertheless, stronger CNN architectures such as VGG-M and VGG-VD 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 $\sim 11-12\%$ 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 5.4. Second, *the advantage of using domain-specific CNNs disappears*. In fact, the same CAFFE model that is 6.4% worse than PLACES with

[2]Zhou et al., 2014 report 68.3% for PLACES applied to MIT Indoor, a small difference explained by implementation details such as the fact that, for all the methods, we do not perform data augmentation by jittering.

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

Table 5.5: **Comparing late and early transfer on the MIT indoor dataset.** We compare PLACES network, trained on MIT Places dataset (similar domain), with a network with the same architecture (AlexNet), trained on a generic domain. We also show results for the late (FC-CNN) and early (FV-CNN) transfer for VGG-M and VGG-VD networks

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 localised 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 our results on MIT Indoor, for both VGG-M and VGG-VD are markedly better (68.8% vs 74.2% / 81.0% resp.) and marginally better (69.7% – Table 5.4 and 5.5) when the same AlexNet CNN is used. Also, when using VLAD instead FV for pooling the convolutional layer descriptors, the performance of our method is still better (68.8% vs 71.2%), as seen in Table 5.3. 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.

Conclusions. Throughout this set of experiments, we are evaluating on MIT Indoors dataset; when pretraining the CNN on MIT Places, which is also a scene understanding dataset, we see an advantage in the case of the late transfer, but this is reduced by early transfer, and even outperformed, surprisingly, when we transfer from a generic domain (ImageNet). Using a better trained network (such as VGG-M) or a deeper architecture (such as VGG-VD) increases the gap even further.

Chapter 6

Semantic Segmentation

In the previous chapter we considered the problem of recognizing textures given the image regions that contain them. In this chapter, we explore instead the problem of automatically recognizing as well as segmenting such textured regions in the image.

6.1 Experimental Setup

Motivated by our previous work [Cimpoi et al., 2014], in which we successfully ported object description methods to texture descriptors, here we propose a segmentation technique inspired by 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.

In this thesis, we explore 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 colour 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 partially contradicts 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 colour 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 labelling. 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.

6.2 Analysis

Results are reported in Table 6.1. Two datasets are evaluated: OS for material recognition and MSRC for things & stuff. Compared to OS+R, classifying crisp regions results in a drop of about 10% per-pixel classification accuracy for all de-

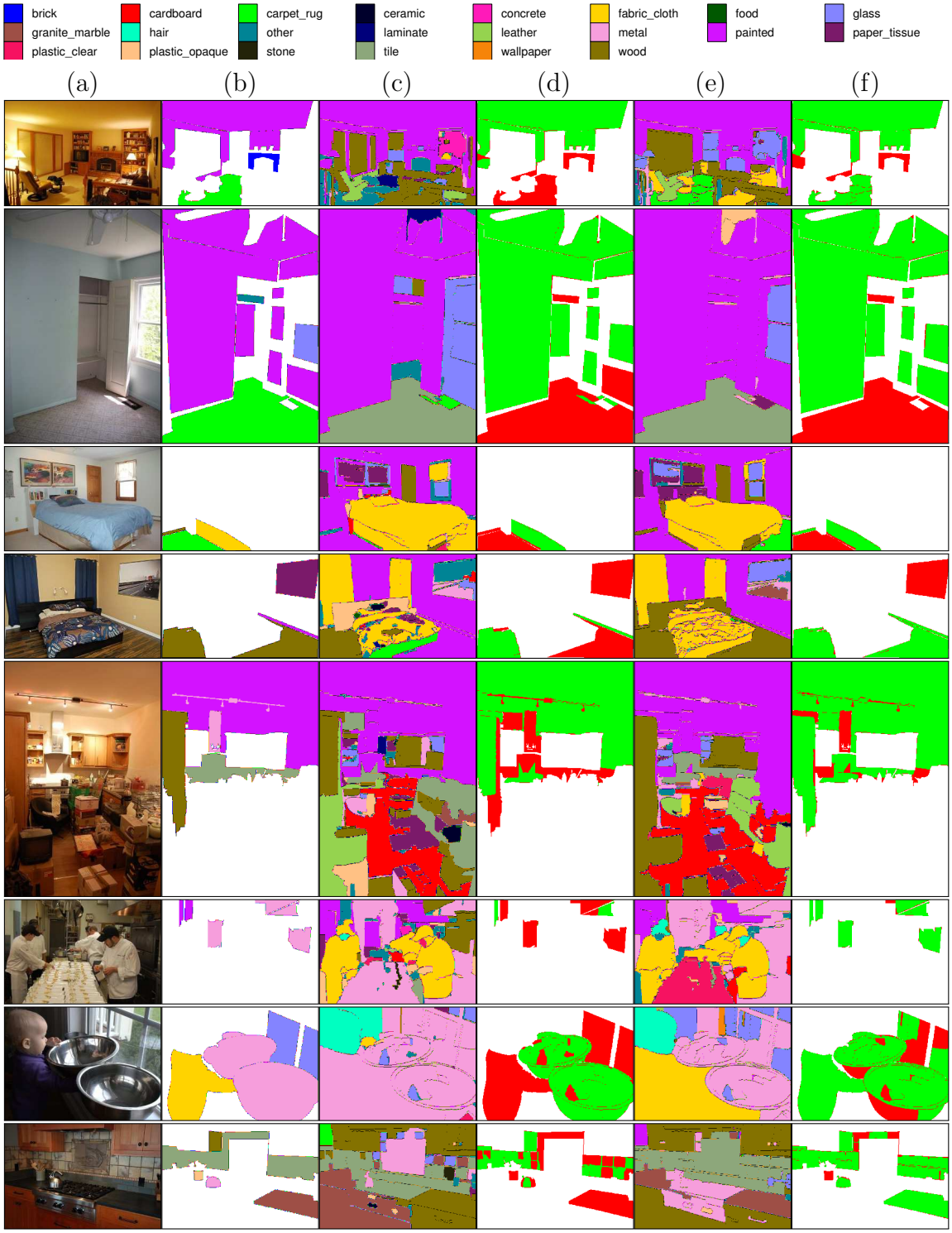


Figure 6.1: **OS material recognition results.** Example test image with material recognition and segmentation on the OS dataset. (a) original image. (b) ground truth segmentations from the OpenSurfaces repository (note that not all pixels are annotated). (c) FC-CNN and crisp-region proposals segmentation results. (d) incorrectly predicted pixels (restricted to the ones annotated). (e-f) the same, but for FV-CNN.

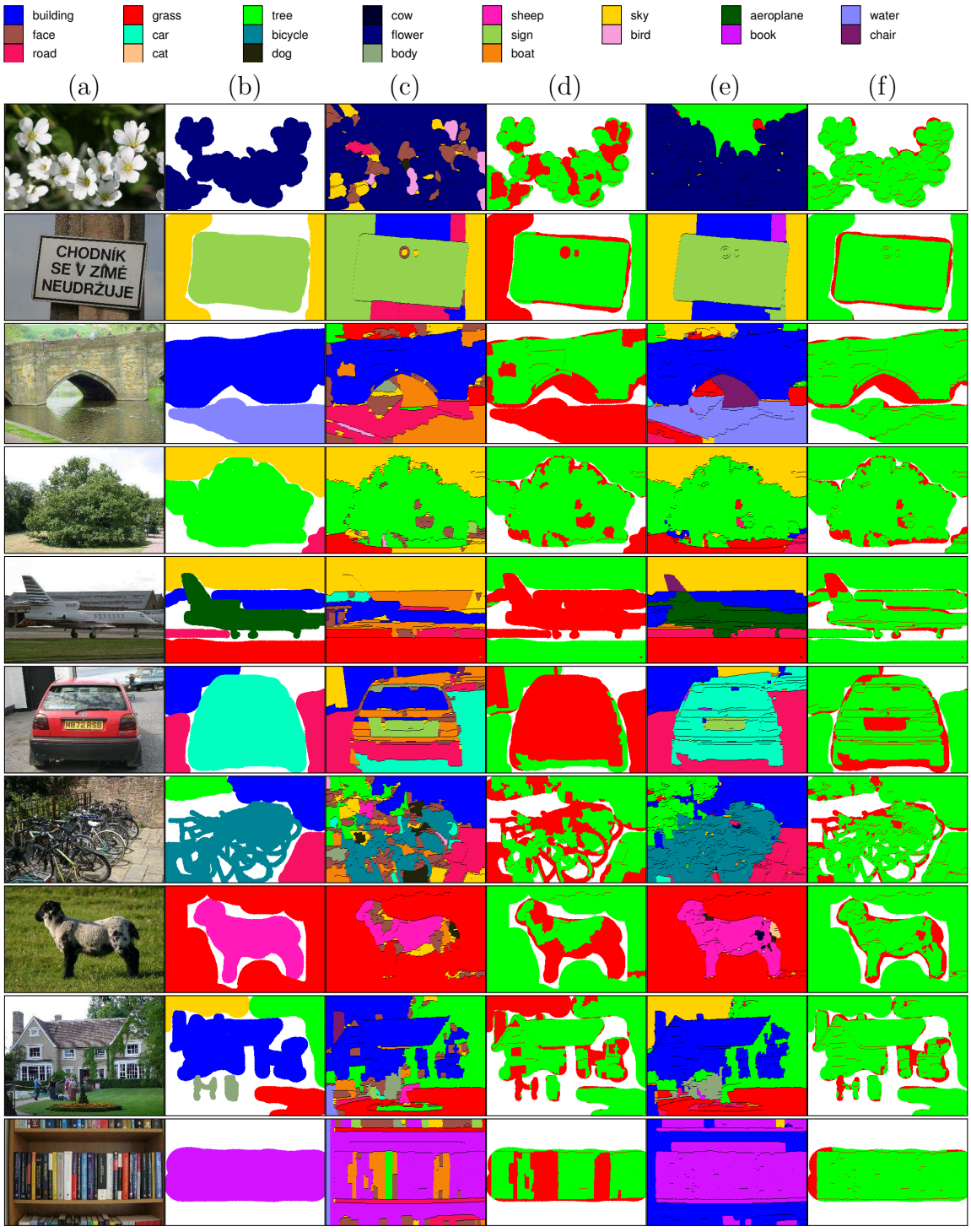


Figure 6.2: **MSRC object segmentation results.** (a) image, (b) ground-truth, (c-d) FC-CNN segmentation and errors, (d-e) FV-CNN segmentation and errors.

	meas.	VGG-M			VGG-VD			
dataset	(%)	FC	FV	FV+FC	FC	FV	FC+FV	SoA
OS	pp	36.0	48.6 (46.9)	49.8	38.5	55.5 (55.7)	55.9	–
OSA	msrc	42.8	66.0	63.4	42.1	67.9	64.6	–
MSRC	msrc	56.1	82.3	75.2	57.7	86.9	81.5	86.5*

Table 6.1: Segmentation and recognition using crisp region proposals of materials (OS) and things & stuff (MSRC). Per-pixel accuracies are reported, using the MSRC variant (see text) for the MSRC dataset. Results using MCG proposals of Arbeláez et al., 2014 are reported in brackets for FV-CNN. * obtained by Ladicky et al., 2010.

scriptors. As this dataset is fairly challenging with best achievable performance is 55.4%, this is a satisfactory result. 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% of 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. Qualitative segmentation results (sampled at random) are given in Figure 6.1 and 6.2.

Results using FV-CNN shown in Table 6.1 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 localised and highly irregularly shaped in an image.

While for recognizing textures, materials or objects covering the entire image, FC-CNN and FV-CNN were similar in performance, the latter consisting in evaluating few layers less, the advantage of FV-CNN becomes obvious for segmentation tasks, as FC-CNN requires recomputing the features for every region proposal.

Chapter 7

Applications of Describable Texture Attributes

This chapter explores two applications of the DTD attributes: using them as a general-purpose texture descriptors (Section 7.1) for material recognition tasks, and as a tool for search and visualisation in specialized catalogues of fabrics or wallpapers (Section7.2).

7.1 Describable Attribute as Generic Texture Descriptors

In this section we evaluate using the 47 describable attributes from Section 3.1 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 descriptors (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.

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	70.81±1.39	73.09±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	**74.52±2.31**	**77.14±1.36**	69.21±1.77	72.17±1.66
Previous best	76.0 ±2.9		57.7±1.7	

Table 7.1: **DTD for material recognition.** Trained on FV-SIFT and FC-CNN features from VGG Very Deep 19 Layers net, the DTD attributes are KTH-T2b and FMD compared to published state of the art results in term of classification accuracy. See the text for the details on the notation and the methods.

Results are reported in Table 7.1 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 colour, 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.

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

Classifier	KTH-T2b		FMD	
Method	Linear	RBF	Linear	RBF
DTD FC-CNN	$70.66_{\pm 1.97}$	$71.16_{\pm 2.38}$	$61.44_{\pm 1.43}$	$63.90_{\pm 1.31}$
ALOT FC-CNN	$70.03_{\pm 0.29}$	$71.37_{\pm 2.04}$	$66.73_{\pm 2.27}$	$69.27_{\pm 1.60}$
FC-CNN VD	$72.30_{\pm 1.68}$	$74.24_{\pm 2.97}$	$69.63_{\pm 1.58}$	$71.09_{\pm 1.73}$
ALOT-FC-CNN-VD	$73.79_{\pm 2.94}$	$74.80_{\pm 2.69}$	$74.46_{\pm 1.84}$	$75.53_{\pm 1.61}$
DTD + ALOT FC-CNN	$72.59_{\pm 0.87}$	$73.02_{\pm 2.64}$	$67.37_{\pm 1.56}$	$69.83_{\pm 1.66}$

Table 7.2: Comparison of DTD and ALOT-based mid-level features for material recognition. The FC-CNN features were used to train 47 and 250 classifiers, respectively.

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% [Sulc and Matas, 2014]. On FMD the DTD features outperform the state-of-the-art by a significant 15% margin: 72.17% accuracy vs. 57.7% [Qi et al., 2014], or 57.1% [Sharan et al., 2013], using multiple features.

In Table 7.2, we show a comparison between the mid-level descriptors based on DTD (47-D) and ALOT (250-D). Similarly to the previous experiment, we computed the scores obtained on images from FMD and KTH-T2b by classifiers trained on DTD and ALOT datasets, respectively. It is interesting to note that, despite 5 times larger, the ALOT descriptor gives similar performance on KTH-T2b. However, there is an approximately 5% improvement showing consistently on FMD.

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 obtained using LBP^u (63.3%). However, LBP^u was developed specifically on the Outex data, and it may be overfitted to this case. To verify this, 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

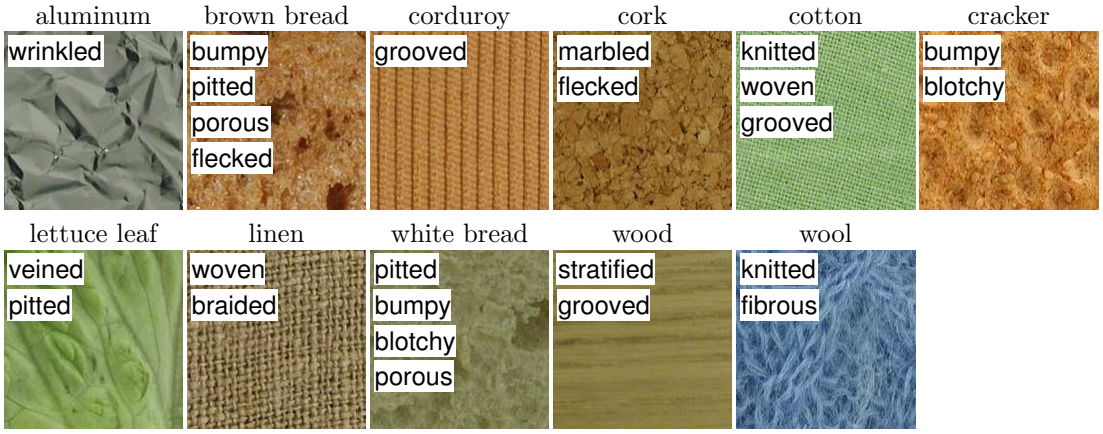


Figure 7.1: Descriptions of materials from KTH-T2b dataset. These words are the most frequent top scoring texture attributes (from the list of 47 we proposed), when classifying the images from the KTH-T2b dataset.

results in an accuracy of 64.5% mAP; at the same time,Table 5.1 shows that LBP^u is not a competitive predictor on DTD. Hence, it is not a limitation of the DTD attributes, but rather the very good performance of LBP^u on Outex that makes the difference in this case. This is remarkable considering that their retrieval experiment contains the data used to *train* their own attributes (target set), while our attributes are trained on a completely different data source.

7.2 Search and Visualisation

This section includes a short qualitative evaluation of the DTD attributes. Perhaps their most appealing property is interpretability; to verify that semantics transfers in a reasonable way across domains, Figure 7.1 shows an excellent semantic correlation between the ten categories in KTH-T2b and the attributes in DTD. For example, aluminium 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 Section 5.1.4 using the FV-SIFT rep-

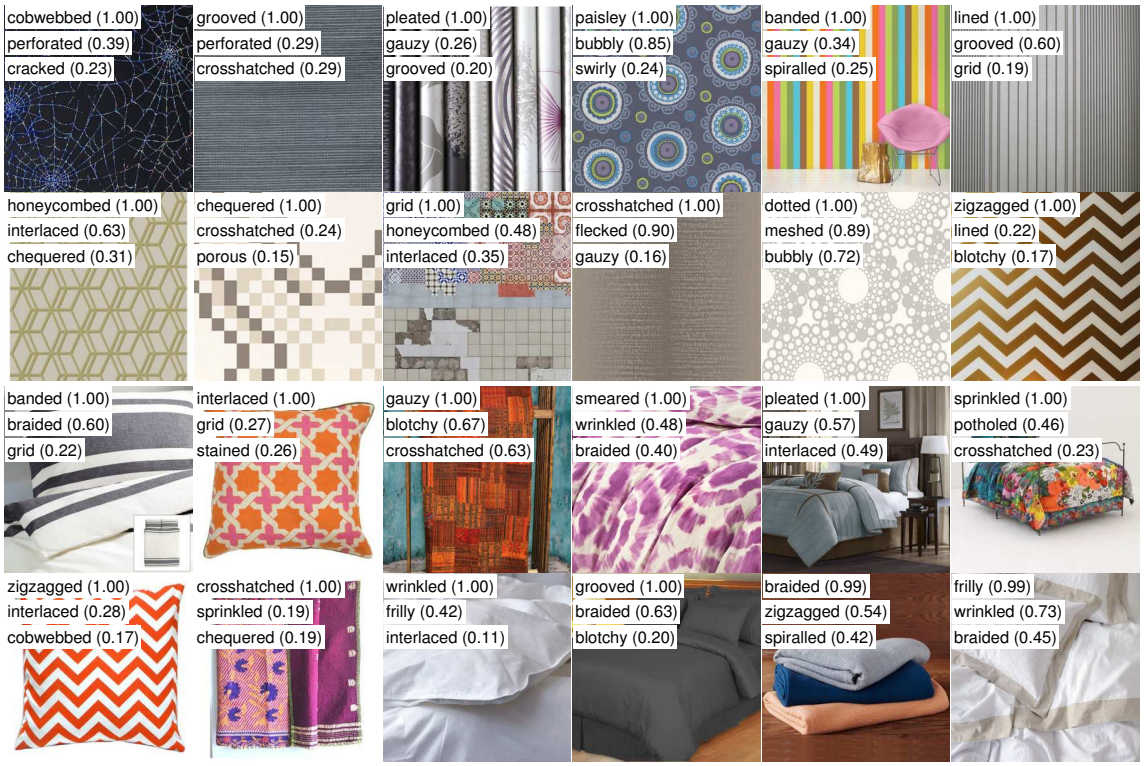


Figure 7.2: Bedding sets (top two rows) and wallpapers (bottom two rows) with the top 3 attributes predicted by our classifier and normalized classification score in brackets.

resentation and them apply them to the 10,000 images to predict the strength of association of each attribute and image. Classifiers scores are re-calibrated on the target data and converted to probabilities by examining the extremal statistics of the scores. Figure 7.2 shows some example attribute predictions, selecting for a number of attribute an image that would score perfectly (excluding images used for calibrating the scores), and then including additional top two attribute matches. The top two matches tend to be very good description of each texture or pattern, while the third is a good match in about half of the cases.

Chapter 8

Conclusions and Future Work

In this chapter we summarise the achievements of the work presented in this thesis, and provide directions for future research.

8.1 Achievements

In this thesis we made several contributions to texture and material recognition. First, we added a new dimension to the problem of recognizing textures, by proposing a rich, human-centric vocabulary of texture attributes. We contributed a benchmark dataset representative of this task by collecting a large number of images "in the wild", jointly labelled with 47 describable texture attributes. We used this dataset to study the problem of extracting semantic properties of textures and patterns.

We also proposed for the first time a large scale analysis of material and texture attribute recognition in clutter. The benchmark for this task was derived from Open Surfaces, an earlier contribution of the computer graphic community, which we also augmented with annotations of semantic attributes. By introducing this new problem of recognizing attributes and materials of textures in the wild and in clutter, we brought the study of textures closer to real-world, human-centric applications.

On the technical side, we analysed texture representations 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 double as a scene and object descriptor too, and resulting in the new state-of-the-art performance in several benchmarks. We conducted a thorough evaluation of existing methods, comparing them with the novel representation we introduced, of which we explored in detail the space of parameters and configurations.

With the proposed texture representations we advanced the state-of-the-art for texture recognition by up to **25%** on challenging benchmarks such as Flickr Material Dataset. In this dataset, we have halved the gap between the recognition performance of computer vision systems and humans, from 57.7% vs 92.4% [Sharan et al., 2014] to 82.4% vs 92.4%. When the exposure of human subjects is limited to 40ms, automatic recognition is of only **2.5%** worse than humans (82.4% vs 84.9%).

We have also shown that the new representations, although designed for texture recognition, generalize well for other problems, like object classification – achieving results comparable to the current state-of-the-art (Pascal VOC 2007), but without using data augmentation. On indoor scenes from MIT Indoor dataset, we advanced the current state-of-the-art by 10% absolute accuracy increase, also showing that, by using FV-CNN, the advantage of domain-tuned CNNs disappears. Similarly, for fine-grained categorization tasks, we achieve comparable or better results on CUB200-2011, without using part or bounding box information.

We also introduced a low-dimensionality mid-level descriptor, which consists of classifier scores for the 47 attributes. When learned using FV-CNN features on very deep models, we obtain 13% accuracy improvement on FMD, and a result approaching closely the state-of-the-art on KTH-T2b. The low dimensionality allows using to use an RBF kernel on top, which improves the results by a further few percent.

8.2 Future Work

There are several directions in which the present work can be extended in the future. One interesting direction is to explore how recognizing describable attributes could further benefit from using CNNs, by fine-tuning a pre-trained network combined with a pooling encoder such as FV, or by training a model specific for texture attribute recognition. However, the number of samples per category is relatively small compared to the large number of parameters in deep networks, so that this would require significantly extending the available texture datasets.

Although we proved through practical applications that using classifiers learnt on DTD can generate meaningful descriptions of new images, the vocabulary of 47 terms could be further extended, to provide more accurate and detailed descriptors. For example, the extended vocabulary could incorporate material-like terms (*e.g.wood-like, metallic, furry*) and terms to better describe surface reflectance properties, (*transparent, shiny*) or structure (*pebble-like*). DTD was collected with the simplifying assumption that each texture fills the entire image, a limitation that was removed in Open Surfaces for some of the attributes. An extension would be to remove these limitations in the future versions of the DTD benchmark. The annotation work could be simplified by providing the output of our method as a starting point.

We explored the querying catalogues only for a limited domain, namely textiles and wallpapers, showing that the top predictions for a given new image are meaningful and informative. This could be further extended to a description-based retrieval benchmark. Inspired by Sivic and Zisserman, 2003, recognizing describable properties of textures would enable querying image catalogues or videos more specifically, *e.g.*retrieving frames containing the character with a chequered shirt, or searching an online store for a striped shirt. The set of attributes could be further extended

with (relative) meta-attributes, like density of constituent elements (dense / sparse stripes or dots) and dimension (larger squares for a chequered pattern).

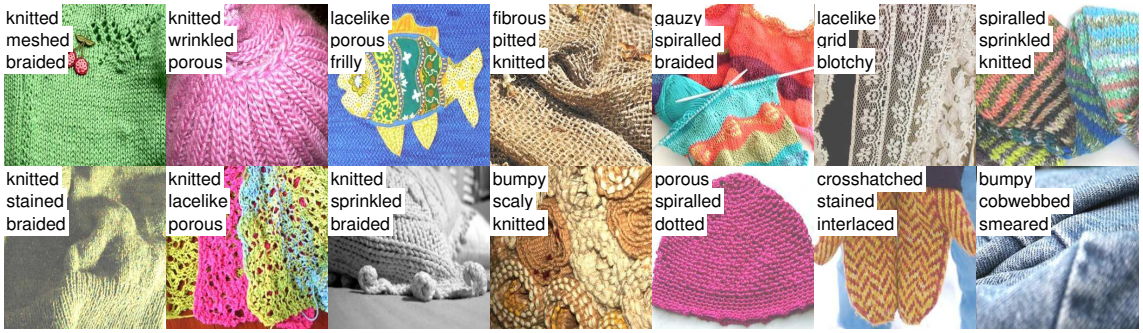
Understanding textures in detail and in a human-centric manner would be easily applicable to graphical editing, providing a more intuitive interface to texture generators, or by allowing to replace a texture with controlled variants of the same, similarly to Liu et al., 2004 that edits near-regular textures.

Appendices

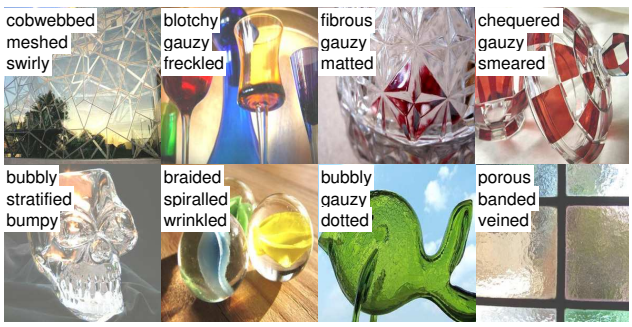
Appendix A

FMD Described

In what follows, we show some more visualizations of the top scoring (three) DTD attributes, recognised on images from FMD dataset.



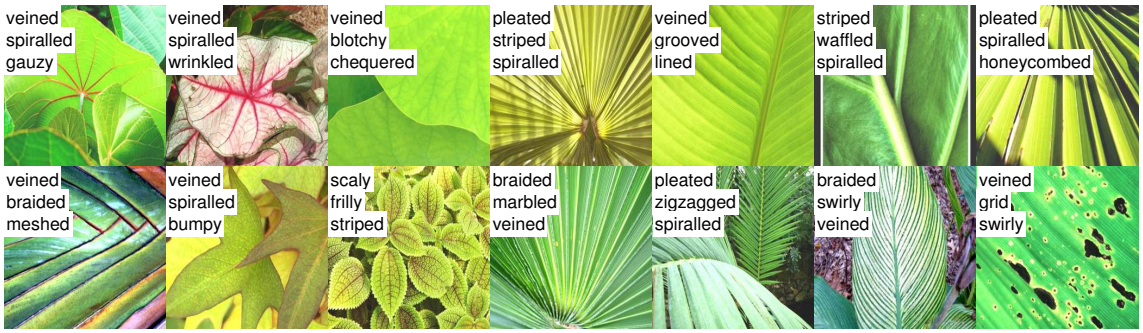
fabric (knitted)



glass (bubbly, gauzy)

glass (crystalline, bubbly)

Figure A.1: Example meaningful clusters of FMD categories, obtained using K-means on DTD classification scores. Showing results for fabric and glass – overlaid, we list the most frequently identified attributes. On each image, we show the top 3 scoring texture words.



foliage (pleated, spiralled, veined)

foliage (frilly, sprinkled)

paper (wrinkled, pleated)

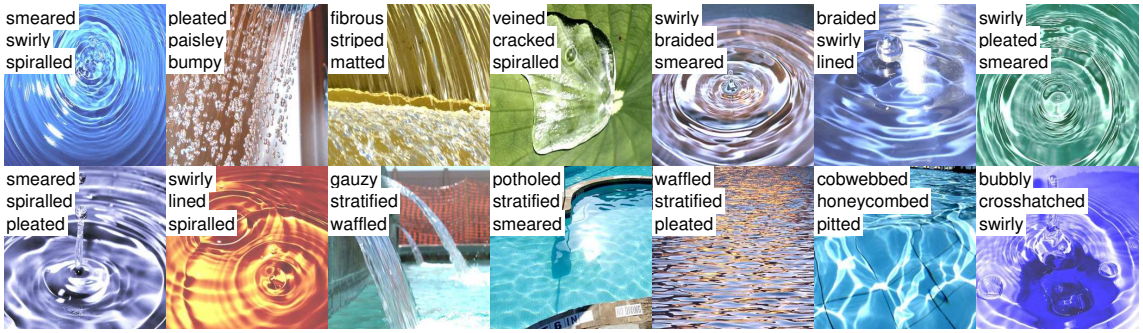
wood (cracked, veined, interlaced)

Figure A.2: *Continued from Figure A.1.* Displaying results on foliage, paper and wood.

stone (bumpy)

stone (porous, pitted, flecked)

water (bubbly, smeared)



water (swirly, spiralled)

Figure A.3: *Continued from Figure A.2* Subcategories for stone and water images.

Appendix B

Describable Textures Dataset

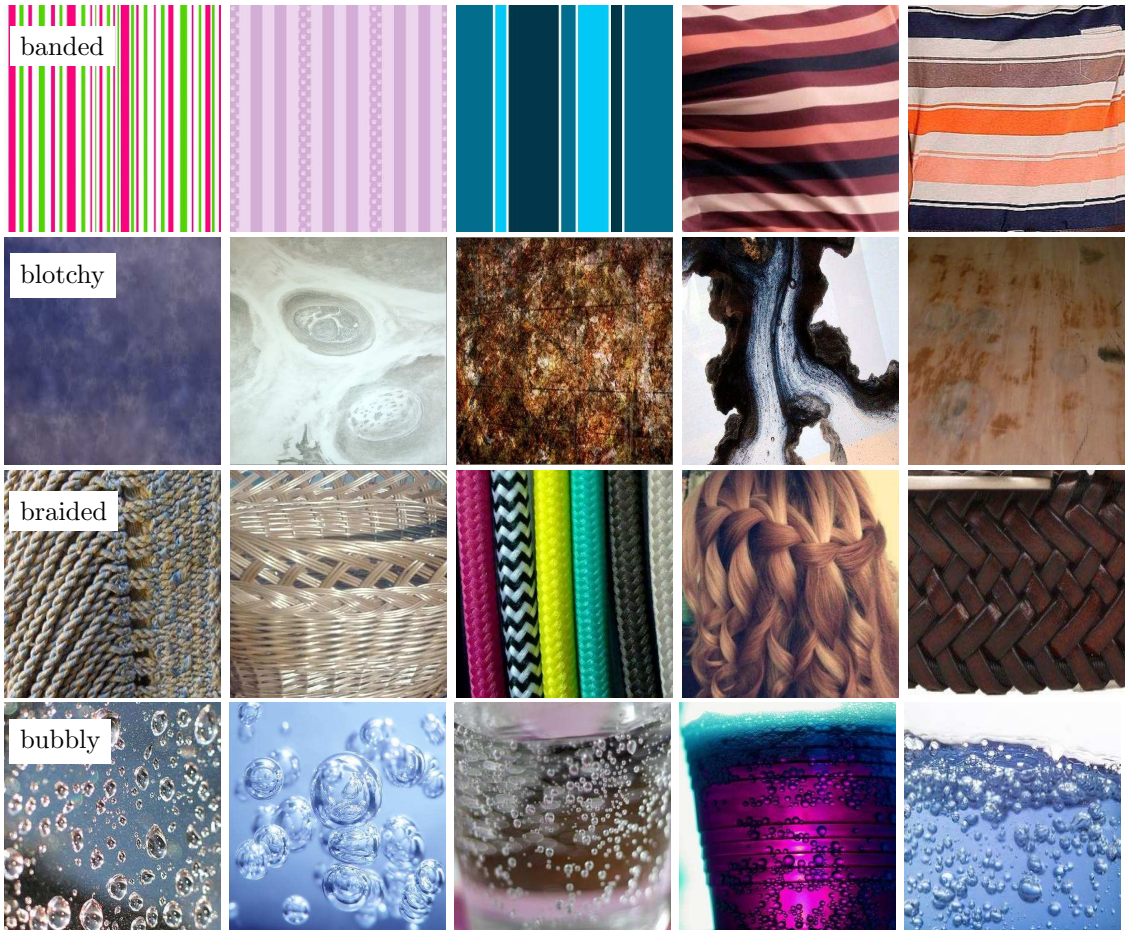


Figure B.1: The 47 texture words in the **Describable Texture Dataset** introduced in Chapter 3. Each row represents sample images for one attribute category.

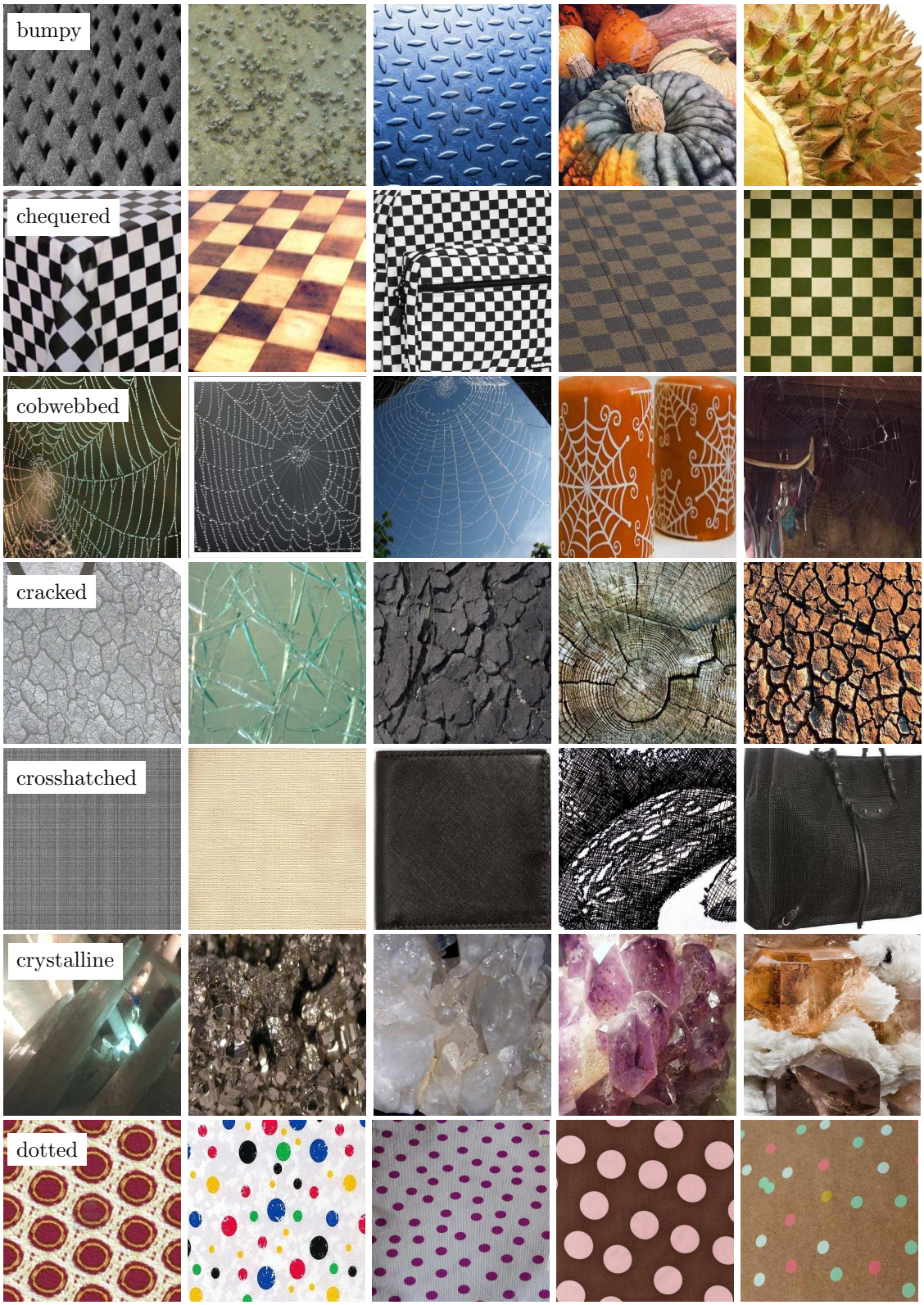


Figure B.2: Describable Textures Dataset *continued from Figure B.1*

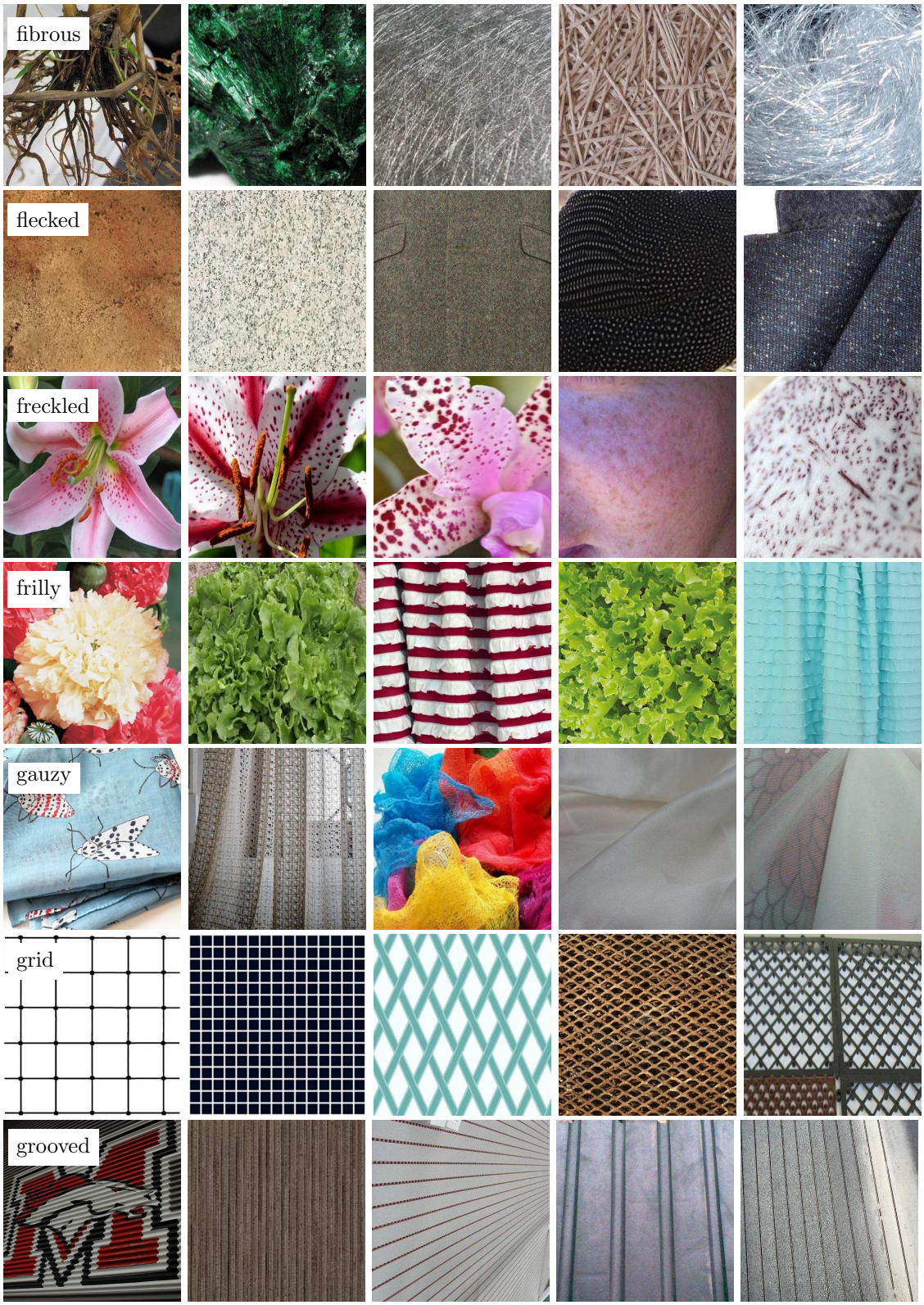


Figure B.3: Describable Textures Dataset *continued from Figure B.2*

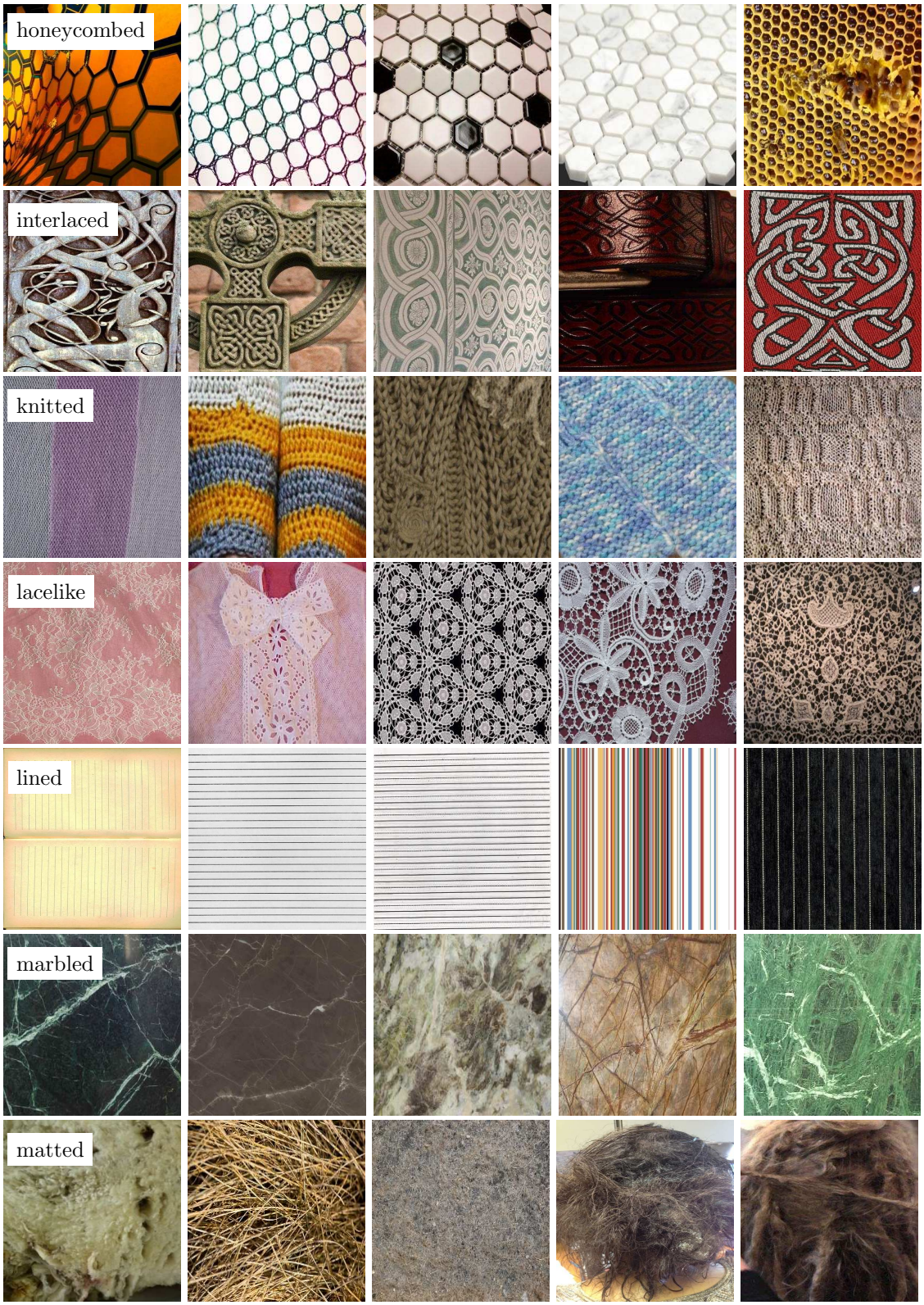


Figure B.4: Describable Textures Dataset *continued from Figure B.3*

Figure B.5: Describable Textures Dataset *continued from Figure B.4*

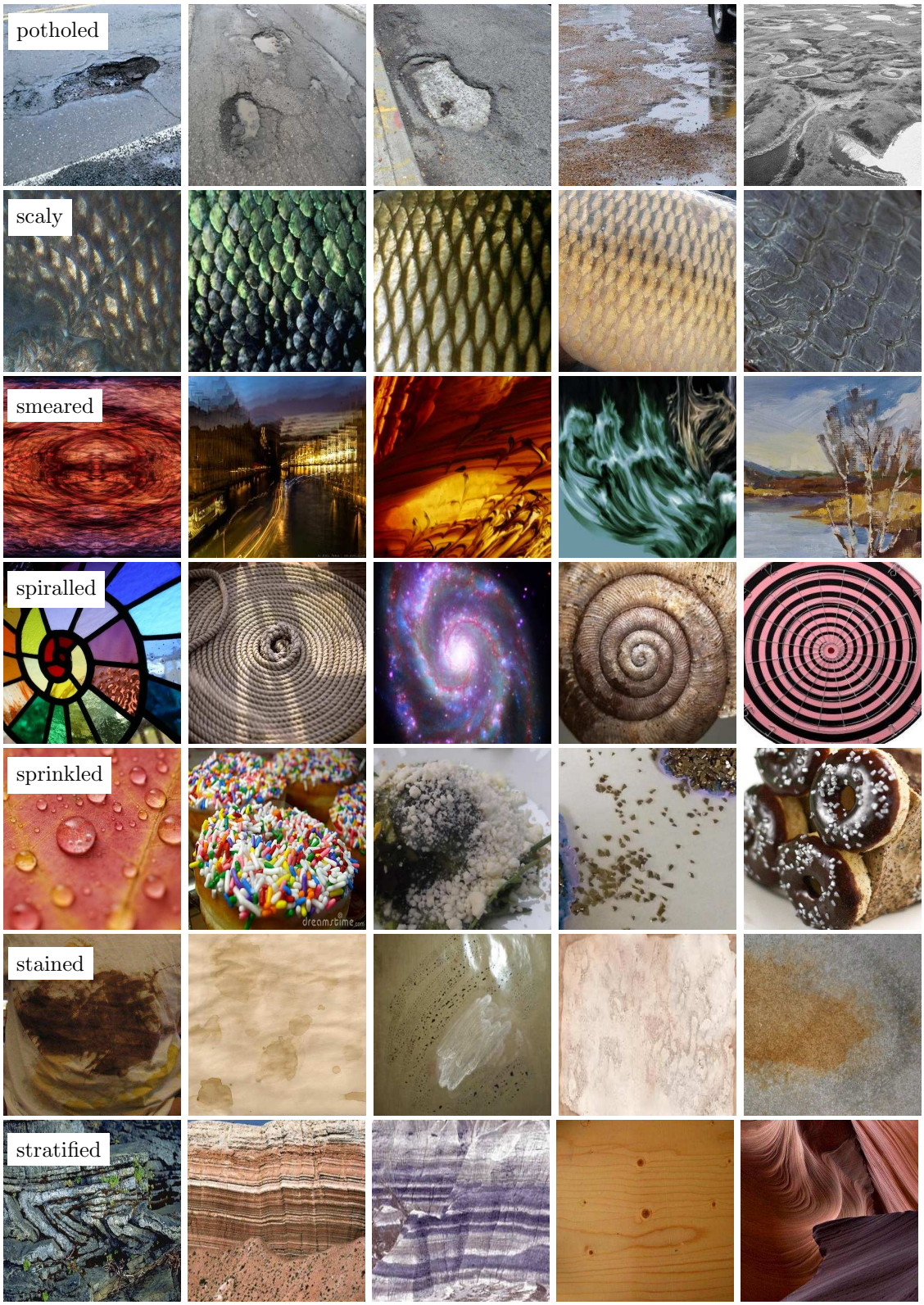


Figure B.6: Describable Textures Dataset *continued from Figure B.5*

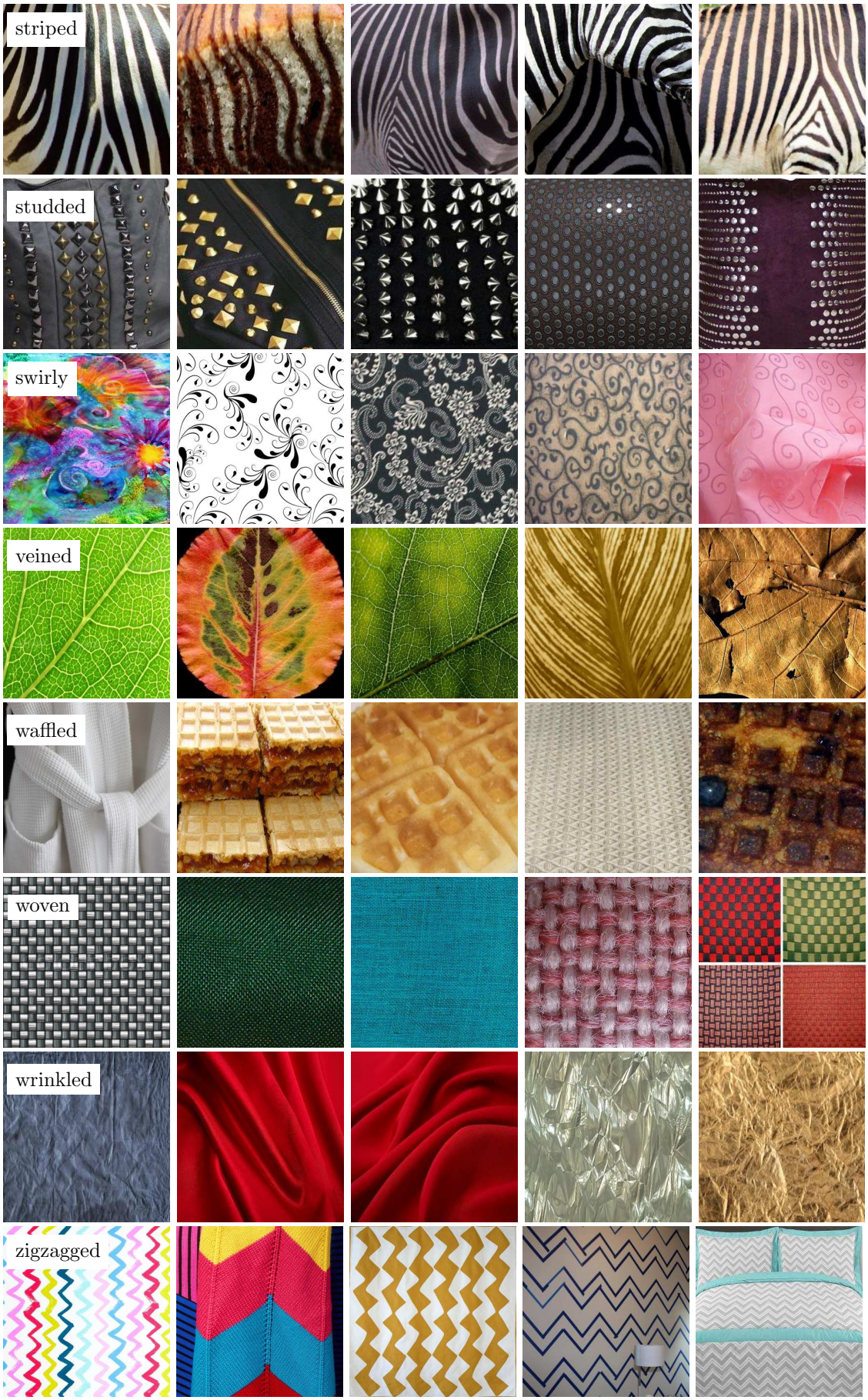


Figure B.7: Describable Textures Dataset *continued from Figure B.6*

References

Adelson, E. H. (2001). "On Seeing Stuff: The Perception of Materials by Humans and Machines". In: *SPIE* 4299.

Amadasun, M. and R. King (1989). "Textural features corresponding to textural properties". In: *Systems, Man, and Cybernetics* 19.5.

Amit, Y., D. Geman, and K. Wilder (1997). "Joint induction of shape features and tree classifiers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.11, pp. 1300–1305.

Arandjelovic, R. and A. Zisserman (2012). "Three things everyone should know to improve object retrieval". In: *Proc. CVPR*.

Arbeláez, P., J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik (2014). "Multiscale Combinatorial Grouping". In: CVPR.

Bach, F., G. Lanckriet, and M. Jordan (2004). "Multiple kernel learning, conic duality and the smo algorithm". In: *Proceedings of the International Conference on Machine Learning*.

Badri, H., H. Yahia, and K. Daoudi (2014). "Fast and Accurate Texture Recognition with Multilayer Convolution and Multifractal Analysis". In: *Proc. ECCV*. Vol. 8689, pp. 505–519.

Bell, S., P. Upchurch, N. Snavely, and K. Bala (2013). "OpenSurfaces: A Richly Annotated Catalog of Surface Appearance". In: *Proc. SIGGRAPH*.

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

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

Bhushan, N., A. Rao, and G. Lohse (1997). "The Texture Lexicon: Understanding the Categorization of Visual Texture Terms and Their Relationship to Texture Images". In: *Cognitive Science* 21.2, pp. 219–246.

Blaschko, M., R. B. Girshick, J. Kannala, I. Kokkinos, S. Mahendran, S. Maji, S. Mohamed, E. Rahtu, N. Saphra, K. Simonyan, B. Taskar, A. Vedaldi, and D. Weiss (2012). *Towards a detailed understanding of objects and scenes in natural images, CLSP 2012 Summer Workshop*. Tech. rep. URL: `http://www.clsp.jhu.edu/user_uploads/ws12/vedaldi/clsp12-tduosn-closing.pdf`.

Breiman, L. (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32.

Brodatz, P. (1966). *Textures: A Photographic Album for Artists & Designers*. New York: Dover.

Brown, M., G. Hua, and S. Winder (2011). "Discriminative Learning of Local Image Descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.1, pp. 43–57.

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

Caputo, B., E. Hayman, and P. Mallikarjuna (2005). "Class-Specific Material Categorisation". In: *Proceedings of the International Conference on Computer Vision*, pp. 1597–1604.

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

Chatfield, K., K. Simonyan, A. Vedaldi, and A. Zisserman (2014). "Return of the Devil in the Details: Delving Deep into Convolutional Nets". In: *Proc. BMVC*.

Chaudhuri, B. and N. Sarkar (1995). "Texture segmentation using fractal dimension". In: *PAMI* 17.1, pp. 72–77.

Cimpoi, M., S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi (2014). "Describing Textures in the Wild". In: *Proc. CVPR*.

Cimpoi, M., S. Maji, and A. Vedaldi (2015). "Deep Filter Banks for Texture Recognition and Description". In: *Proc. CVPR*.

Clarke, A. D. F., F. Halley, A. J. Newell, L. D. Griffin, and M. J. Chantler (2011). "Perceptual Similarity: A Texture Challenge". In: *BMVC*.

Cortes, C. and V. Vapnik (1995). "Support-Vector Networks". In: *Machine Learning* 20.3, pp. 273–297. URL: `citeseer.nj.nec.com/cortes95supportvector.html`.

Csurka, G., C. R. Dance, L. Dan, J. Willamowski, and C. Bray (2004). "Visual Categorization with Bags of Keypoints". In: *Proc. ECCV Workshop on Stat. Learn. in Comp. Vision*.

Cula, O. G. and K. J. Dana (2001). "Compact Representation of Bidirectional Texture Functions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1041–1047.

Dana, K. J. and S. Nayar (1998). "Histogram model for 3d textures". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 618–624.

Dana, K. J., B. van Ginneken, S. K. Nayar, and J. J. Koenderink (1999). "Reflectance and Texture of Real World Surfaces". In: *ACM Transactions on Graphics* 18.1, pp. 1–34.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR*.

Donahue, J., Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell (2013). "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *Proc. ICML*.

Dunn, D., W. Higgins, and J. Wakeley (1994). "Texture segmentation using 2-D Gabor elementary functions". In: *PAMI* 16.2, pp. 130–149.

Efros, A. and T. Leung (1999). "Texture synthesis by non-parametric sampling". In: *CVPR*. Vol. 2.

Everingham, M., A. Zisserman, C. Williams, and L. V. Gool (2007). *The PASCAL Visual Obiect Classes Challenge 2007 (VOC2007) Results*. Tech. rep. Pascal Challenge.

Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman (2008). *The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results*. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

Farhadi, A., I. Endres, D. Hoiem, and D. Forsyth (2009). "Describing Objects by Their Attributes". In: *Proc. CVPR*.

Farhadi, A., I. Endres, and D. Hoiem (2010). "Attribute-Centric Recognition for Cross-category Generalization". In: *Proc. CVPR*.

Ferrari, V. and A. Zisserman (2007). "Learning Visual Attributes". In: *Proc. NIPS*.

Forsyth, D. (2001). "Shape from texture and integrability". In: *ICCV*. Vol. 2. IEEE, pp. 447–452.

Gårding, J. (1992). "Shape from texture for smooth curved surfaces in perspective projection". In: *Journal of Mathematical Imaging and Vision* 2.4, pp. 327–350.

Geusebroek, J. M., A. W. M. Smeulders, and J. van de Weijer (2003). "Fast Anisotropic Gauss Filtering". In: *IEEE Transactions on Image Processing* 12.8, pp. 938–943.

Girshick, R. B., J. Donahue, T. Darrell, and J. Malik (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proc. CVPR*.

Gong, Y., L. Wang, R. Guo, and S. Lazebnik (2014). "Multi-Scale Orderless Pooling of Deep Convolutional Activation Features". In: *Proc. ECCV*.

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

Hayman, E., B. Caputo, M. Fritz, and J.-O. Eklundh (2004). "On the significance of real-world conditions for material classification". In: *Computer Vision–ECCV 2004*.

He, K., X. Zhang, S. Ren, and J. Sun (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Proc. ECCV*.

Isola, P., D. Zoran, D. Krishnan, and E. H. Adelson (2014). "Crisp Boundary Detection Using Pointwise Mutual Information". In: *Proc. ECCV*.

Jain, A. and F. Farrokhnia (1991). "Unsupervised texture segmentation using Gabor filters". In: *Pattern recognition* 24.12, pp. 1167–1186.

Jégou, H., M. Douze, C. Schmid, and P. Pérez (2010). "Aggregating local descriptors into a compact image representation". In: *Proc. CVPR*.

Jia, Y. (2013). *Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding*. `http://caffe.berkeleyvision.org/`.

Joachims, T. (2002). "Optimizing Search Engines Using Clickthrough Data". In: *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, pp. 133–142.

Julesz, B. (1962). "Visual pattern discrimination". In: *IRE Transactions on Information theory, IT-8 84-92.*

Julesz, B. (1981). "Textons, the Elements of Texture Perception, and their Interactions". In: *Nature* 290, pp. 91–97.

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

Kovashka, A., D. Parikh, and K. Grauman (2012). "WhittleSearch: Image Search with Relative Attribute Feedback". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proc. NIPS*.

Ladicky, L., C. Russell, P. Kohli, and P. Torr (2010). "Graph Cut Based Inference with Co-occurrence Statistics". In: *Proc. ECCV*, pp. 239–253.

Lampert, C. H., H. Nickisch, and S. Harmeling (2009). "Learning to detect unseen object classes by between-class attribute transfer". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 951–958.

Lazebnik, S., C. Schmid, and J. Ponce (2005). "A Sparse Texture Representation Using Local Affine Regions". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.8, pp. 2169–2178.

Lazebnik, S., C. Schmid, and J Ponce (2006). "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4, pp. 541–551.

Leung, T. and J. Malik (1999). "Recognizing surfaces using three-dimensional textons". In: *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*. Kerkyra, Greece, pp. 1010–1017.

Leung, T. and J. Malik (2001). "Representing and recognizing the visual appearance of materials using three-dimensional textons". In: *International Journal of Computer Vision* 43.1, pp. 29–44.

Liu, C., L. Sharan, E. H. Adelson, and R. Rosenholtz (2010). "Exploring Features in a Bayesian Framework for Material Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

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

Liu, Y., W.-C. Lin, and J. Hays (2004). "Near-regular texture analysis and manipulation". In: *ACM Transactions on Graphics (TOG)*. Vol. 23. 3. ACM, pp. 368–376.

Lowe, D. G. (1999). "Object Recognition from Local Scale-Invariant Features". In: *Proc. ICCV*.

Mairal, J., F. Bach, J. Ponce, G. Sapiro, and A. Zisserman (2008). "Supervised Dictionary Learning". In: *Advances in Neural Information Processing Systems*, pp. 1033–1040.

Maji, S., A. C. Berg, and J. Malik (2008). "Classification using Intersection Kernel Support Vector Machines is Efficient". In: *Proc. CVPR*.

Malik, J. and P. Perona (1990). "Preattentive texture discrimination with early vision mechanisms". In: *JOSA A* 7.5.

Malik, J. and R. Rosenholtz (1997). "Computing local surface orientation and shape from texture for curved surfaces". In: *International Journal of Computer Vision* 23.2, pp. 149–168.

Mallikarjuna, P., M. Fritz, A. T. Targhi, E. Hayman, B. Caputo, and J.-O. Eklundh (2005). *The KTH-TIPS and KTH-TIPS2 databases*.

Manjunath, B. and R. Chellappa (1991). "Unsupervised texture segmentation using Markov random field models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.5, pp. 478–482.

Matthews, T., M. S. Nixon, and M. Niranjan (2013). "Enriching Texture Analysis with Semantic Data". In: *CVPR*.

McCallum, A. (1999). "Multi-label text classification with a mixture model trained by EM". In: *AAAI workshop on Text Learning*.

Mikolajczyk, K. and C. Schmid (2005). "A performance evaluation of local descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10, pp. 1615–1630.

Mikolajczyk, K., T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool (2005). "A Comparison of Affine Region Detectors". In: *International Journal of Computer Vision* 65.1/2, pp. 43–72.

Ojala, T., M. Pietikäinen, and D. Harwood (1996). "A Comparative Study of Texture Measures with Classification Based on Feature Distributions". In: *Pattern Recognition* 29.

Ojala, T., M. Pietikainen, and T. Maenpaa (2002). "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns". In: *PAMI* 24.7, pp. 971–987.

Olshausen, B. A. and D. J. Field (1997). "Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?" In: *Vision Research* 37.23, pp. 3311–3325.

Oquab, M., L. Bottou, I. Laptev, and J. Sivic (2014). "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks". In: *Proc. CVPR*.

Oxholm, G., P. Bariya, and K. Nishino (2012). "The Scale of Geometric Texture". In: *European Conference on Computer Vision*. Springer Berlin/Heidelberg, pp. 58–71.

Parikh, D. and K. Grauman (2011). "Relative Attributes". In: *Proc. ICCV*.

Parkhi, O. M., K. Simonyan, A. Vedaldi, and A. Zisserman (2014). "A Compact and Discriminative Face Track Descriptor". In: *Proc. CVPR*.

Perronnin, F. and C. R. Dance (2007). "Fisher Kernels on Visual Vocabularies for Image Categorization." In: *CVPR*.

Perronnin, F., J. Sánchez, and T. Mensink (2010). "Improving the Fisher Kernel for Large-Scale Image Classification". In: *Proc. ECCV*.

Perronnin, F. and D. Larlus (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.

Philbin, J., O. Chum, M. Isard, J. Sivic, and A. Zisserman (2008). "Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*.

Platt, J. C. (2000). "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Large Margin Classifiers*. Ed. by A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans. Cambridge.

Portilla, J. and E. Simoncelli (2000). "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International Journal of Computer Vision* 40.1, pp. 49–70.

Qi, X., R. Xiao, C. G. Li, Y. Qiao, J. Guo, and X. Tang (2014). "Pairwise Rotation Invariant Co-Occurrence Local Binary Pattern". In: *PAMI* 36.11, pp. 2199–2213.

Quattoni, A. and A. Torralba (2009). "Recognizing indoor scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rao, A. R. and G. L. Lohse (1996). "Towards a texture naming system: Identifying relevant dimensions of texture". In: *Vision Research* 36.11, pp. 1649 –1669.

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

Schmid, C. (2001). "Constructing models for content-based image retrieval". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2, pp. 39–45.

Schwartz, G. and K. Nishino (2013). "Visual Material Traits: Recognizing Per-Pixel Material Context". In: *Proc. CVCP*.

Sermanet, P., D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun (2013). "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229*.

Sharan, L., R. Rosenholtz, and E. H. Adelson (2009). "Material perceprion: What can you see in a brief glance?" In: *Journal of Vision* 9:784.8.

Sharan, L., C. Liu, R. Rosenholtz, and E. H. Adelson (2013). "Recognizing Materials Using Perceptually Inspired Features". In: *International Journal of Computer Vision* 103.3, pp. 348–371.

Sharan, L., R. Rosenholtz, and E. H. Adelson (2014). "Accuracy and speed of material categorization in real-world images". In: *Journal of Vision* 14.9.

Sharma, G., S. ul Hussain, and F. Jurie (2012). "Local higher-order statistics (LHS) for texture categorization and facial analysis". In: *Proc. ECCV*.

Sifre, L. and S. Mallat (2013). "Rotation, Scaling and Deformation Invariant Scattering for Texture Discrimination". In: *CVPR*.

Simonyan, K. and A. Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556.

Simonyan, K., A. Vedaldi, and A. Zisserman (2012). "Descriptor Learning Using Convex Optimisation". In: *Proceedings of the European Conference on Computer Vision*.

Sivic, J. and A. Zisserman (2003). "Video Google: A Text Retrieval Approach to Object Matching in Videos". In: *Proceedings of the 9th International Conference on Computer Vision, Nice, France*. Vol. 2, pp. 1470–1477.

Sulc, M. and J. Matas (2014). *Fast Features Invariant to Rotation and Scale of Texture*. Tech. rep.

Timofte, R. and L. Van Gool (2012). "A Training-free Classification Framework for Textures, Writers, and Materials". In: *BMVC*.

Tola, E., V. Lepetit, and P. Fua (2008). "A Fast Local Descriptor for Dense Matching". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Varma, M. and A. Zisserman (2002). "Classifying Images of Materials: Achieving Viewpoint and Illumination Independence". In: *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*. Vol. 3. Springer-Verlag, pp. 255–271.

Varma, M. and A. Zisserman (2003). "Texture classification: Are filter banks necessary?" In: *CVPR*. Vol. 2. IEEE, pp. II–691.

Varma, M. and A. Zisserman (2005). "A statistical approach to texture classification from single images". In: *IJCV* 62.1, pp. 61–81.

Varma, M. and A. Zisserman (2009). "A Statistical Approach to Material Classification using Image Patch Exemplars". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.11, pp. 2032–2047.

Vedaldi, A. and B. B. Fulkerson (2008). *VLFeat: An Open and Portable Library of Computer Vision Algorithms*.

Vedaldi, A. and A. Zisserman (2010). "Efficient Additive Kernels via Explicit Feature Maps". In: *CVPR*.

Vedaldi, A. and K. Lenc (2014). "MatConvNet-Convolutional Neural Networks for MATLAB". In: *arXiv preprint arXiv:1412.4564*.

Wah, C., S. Branson, P. Welinder, P. Perona, and S. Belongie (2011). *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology.

Wang, J., J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong (2010). "Locality-constrained linear coding for image classification". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, pp. 3360–3367.

Wang, L. and D.-C. He (1990). "Texture classification using texture spectrum". In: *Pattern Recognition* 23.8.

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

Wei, Y., W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan (2014). "CNN: Single-label to Multi-label". In:

Winder, S., G. Hua, and M. Brown (2009). "Picking the best DAISY". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 178–185.

Xu, Y., H. Ji, and C. Fermuller (2009). "Viewpoint Invariant Texture Description Using Fractal Analysis". In: *IJCV* 83.1, pp. 85–100.

Yang, J., K. Yu, and T. Huang (2010). "Supervised Translation-Invariant Sparse Coding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhang, N., J. Donahue, R. Girshickr, and T. Darrell (2014). "Part-based R-CNNs for Fine-grained Category Detection". In: *Proc. ECCV*.

Zhou, B., A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva (2014). "Learning Deep Features for Scene Recognition using Places Database". In: *Proc. NIPS*.

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