From BoW to CNN: Two Decades of Texture Representation for Texture Classification

Li Liu 1, 2 · Jie Chen 2 · Paul Fieguth 3 · Guoying Zhao 2 · Rama Chellappa 4 · Matti Pietikäinen 2

1 Introduction

Our visual world is richly filled with a great variety of textures, present in images ranging from multispectral satellite data to microscopic images of tissue samples (see Fig. 1). As a powerful visual cue, like color, texture provides useful information in identifying objects or regions of interest in images. Texture is different from color in that it refers to the spatial organization of a set of basic elements or primitives (i.e., textons), the fundamental microstructures in natural images and the atoms of preattentive human visual perception [93]. A textured region will obey some statistical properties, exhibiting periodically repeated textons with some degree of variability in their appearance and relative position [60]. Textures may range from purely stochastic to perfectly regular and everything in between (see Fig. 1).

As a longstanding, fundamental and challenging problem in the fields of computer vision and pattern recognition, texture analysis has been a topic of intensive research since the 1960’s [92] due to its significance both in understanding how the texture perception process works in human vision as well as in the important role it plays in a wide variety of applications. The analysis of texture traditionally embraces several problems including classification, segmentation, synthesis and shape from texture [224]. Significant progress has been made since the 1990’s in the first three areas, with shape from texture receiving comparatively less attention. Typical applications of texture analysis include medical image analysis [48, 158, 177], quality inspection [250], content based image retrieval [148, 216, 266], analysis of satellite or aerial imagery [96, 83], face analysis [4, 49, 215, 265], biometrics [137, 185], object recognition [210, 173, 263], texture synthesis for computer graphics and image compression [65, 66], and robot vision and autonomous navigation for unmanned aerial vehicles. The ever-increasing amount of image and video data due to surveillance, handheld devices, medical imaging, robotics etc. offers an endless potential for further applications of texture analysis.

Texture representation, i.e., the extraction of features that describe texture information, is at the core of texture analysis. After over five decades of continuous research, many kinds of theories and algorithms have emerged, with major surveys and some representative work as follows. The majority of texture features before 1990 can be found in surveys and comparative studies [39, 79, 160, 195, 224, 234, 245]. Tuceryan and Jain [224] identified five major categories of features for texture discrimination: statistical, geometrical, structural, model based, and filtering based features. In 1996, Ojala et al. [161] carried out a comparative study to evaluate the classification performance of several texture features. In 1999, Randen and Husøy [190] reviewed most major filtering based texture features and performed a comparative performance evaluation for texture segmentation. In 2002, Zhang and Tan [262] reviewed invariant texture feature extraction methods. In 2007, Zhang et al. [263] evaluated the performance of several major invariant local texture descriptors. The 2008 book ‘Hand-
book of Texture Analysis” edited by Mirmehdi et al. [157] contains representative work on texture analysis — from 2D to 3D, from feature extraction to synthesis, and from texture image acquisition to classification. The book “Computer Vision Using Local Binary Patterns” by Pietikäinen et al. [185] in 2011 provides an excellent overview of the theory of Local Binary Patterns (LBP) and the use in solving various kinds of problems in computer vision, especially in biomedical applications and biometric recognition systems. Huang et al. [86] presented a review of the LBP variants in the application area of facial image analysis. The book “Local Binary Patterns: New Variants and Applications” by Brahnam et al. [21] in 2014 is a collection of several new LBP variants and their applications to face recognition. More recently, Liu et al. [132] conducted a taxonomy of recent LBP variants and performed a large scale performance evaluation of forty texture features. Researchers [189, 5] presented a review of exemplar based texture synthesis approaches.

The published surveys [39, 79, 160, 194, 195, 161, 183, 224, 234] mainly reviewed or compared methods prior to 1995. Similarly, the articles [190, 262] only covered approaches before 2000. There are more recent surveys [21, 86, 132, 185], however they focused exclusively on texture features based on LBP. The emergence of many powerful texture analysis techniques has given rise to a further increase in research activity in texture research since 2000, however none of these published surveys provides an extensive survey over that time. Given recent developments, we believe that there is a need for an updated survey, motivating this present work. A thorough review and survey of existing work, the focus of this paper, will contribute to more progress in texture analysis. Our goal is to overview the core tasks and key challenges in texture representation approaches, to define taxonomies of representative approaches, to provide a review of texture datasets, and to summarize the performance of the state of the art on publicly available datasets. According to the different visual representations, this survey categorizes the texture representation literature into three broad types: BoW-based, CNN-based, and attribute-based. The BoW-based methods are organized according to their key components. The CNN-based methods are categorized into one of pretrained CNN models, finetuned CNN models, or handcrafted deep convolutional networks.

The remainder of this paper is organized as follows. Related background, including the problem and its applications, the progress made during the past decades, and the challenges of the problem, are summarized in Section 2. From Sections 3 to 5 we give a detailed review of texture representation techniques for texture classification by providing a taxonomy to more clearly group the prominent alternatives. A summarization of benchmark texture databases and state of the art performance is given in Section 6. Section 7 concludes the paper with a discussion of promising directions for texture representation.

2 Background

2.1 The Problem

Texture analysis can be divided into four areas: classification, segmentation, synthesis, and shape from texture [224]. Texture classification [110, 125, 224, 236, 237] deals with designing algorithms for declaring a given texture region or image as belonging to one of a set of known texture categories of which training samples have been provided. Texture classification may also be a binary hypothesis testing problem, such as differentiating a texture as being within or outside of a given class, such as distinguishing between healthy and pathological tissues in medical image analysis. The goal of texture segmentation is to partition a given image into disjoint regions of homogeneous texture [89, 147, 194, 210]. Texture synthesis is the process of generating new texture images which are perceptually equivalent to a given texture sample [56, 65, 186, 189, 243, 270]. As textures provide powerful shape cues, approaches for shape from texture attempt to recover the three dimensional shape of a textured object from its image. It should be noted that the concept of “texture” may have different connotations or definitions depending on the given objective. Classification, segmentation, and synthesis are closely related and widely studied, with shape from texture receiving comparatively less attention. Nevertheless, texture representation is at the core of these four problems. Texture representation, together with texture classification, will form the primary focus of this survey.

As a classical pattern recognition problem, texture classification primarily consists of two critical subproblems: texture representation and classification [90]. It is generally agreed that the extraction of powerful texture features plays a relatively more important role, since if poor features are used even the best classifier will fail to achieve good results. While this survey is not explicitly concerned with texture synthesis, studying synthesis can be instructive, for example, classification of textures via analysis by
The evolution of texture representation over the past decades (see discussion in Section 2.2).

| Year | Event                                                                 |
|------|----------------------------------------------------------------------|
| 1962 | Texture Perception Model (Bela Julesz)                                |
| 1970 | Fractal Model (Keller et al.)                                        |
| 1980 | Laws Masks (Kenneth Laws)                                             |
| 1985 | Wavelet (Stephane Mallat)                                             |
| 1990 | GLCM (Haralick et al.)                                               |
| 1994 | The Texton Theory (Bela Julesz, Nature)                               |
| 2000 | Unified Theory Modeling (Zhu et al.)                                  |
| 2002 | The Texton Theory                                                     |
| 2006 | Video Google (Sivic and Zisserman)                                    |
| 2010 | CNN for ImageNet (Krizhevsky et al.)                                  |
| 2012 | DeCAF and IFV (Cimpoi et al.)                                        |
| 2014 | DCNN for ImageNet                                                      |
| 2015 | FV-CNN (Cimpoi et al.)                                               |

**Fig. 2**

The importance of texture representations (such as Gabor filters [148], LBP [163], BoT [114], Fisher Vector (FV) [203], and wavelet Scattering Convolution Networks (ScatNet) [24]) is that they were found to be well applicable to other problems of image understanding and computer vision, such as object recognition [57, 201], scene classification [18, 36, 106, 197] and facial image analysis [3, 215, 265]. For instance, recently many of the best object recognition approaches in challenges such as PASCAL VOC [57] and ImageNet ILSVRC [201] were based on variants of texture representations. Beyond BoT [114] and FV [203], researchers developed Bag of Semantics (BoS) [50, 51, 106, 118, 191] which requires classifying image patches using BoT or CNN and considers the class posterior probability vectors as locally extracted semantic descriptors. On the other hand, texture representations...
2.3 Key Challenges

In spite of several decades of development, most texture features have not been capable of performing at a level sufficient for real-world textures and are computationally too complex to meet the real-time requirements of many computer vision applications. The inherent difficulty in obtaining powerful texture representations lies in balancing two competing goals: high quality representation and high efficiency.

**High Quality** related challenges mainly arise due to the large intraclass appearance variations caused by changes in illumination, rotation, scale, blur, noise, occlusion, etc. and potentially small interclass appearance differences, requiring texture representations to be of high robustness and distinctiveness. Illustrative examples are shown in Fig. 3. A further difficulty is in obtaining sufficient training data in the form of labeled examples, which are frequently available only in limited amounts due to collection time or cost.

**High Efficiency** related challenges include the potentially large number of different texture categories and their high dimensional representations. Here we have polar opposite motivations: that of big data, with associated grand challenges and the scalability/complexity of huge problems, and that of tiny devices, the growing need for deploying highly compact and efficient texture representations on resource-limited platforms such as embedded and handheld devices.

---

Fig. 3 Illustrations of challenges in texture recognition. Dramatic intraclass variations: (a) Illumination variations, (b) View point and local nonrigid deformation, (c) Scale variations, and (d) Different instances from the same category. Small interclass variations make the problem harder still: (e) Images from the FMD database, and (f) Images from the LFMD database (photographed with a light-field camera). The reader is invited to identify the material category of the foreground surfaces in each image in (e) and (f). The correct answers are (from left to right): (e) glass, leather, plastic, wood, plastic, metal, wood, and plastic; (f) leather, fabric, metal, metal, paper, leather, water, sky and plastic. Section 6 gives details regarding texture databases.

Fig. 4 The goal of texture representation is to transform the input texture image into a feature vector that describes the properties of the texture, facilitating subsequent tasks such as texture recognition. Usually a texture image is first transformed into a pool of local features, which are then aggregated into a global representation for an entire image or region.

3 Bag of Words based Texture Representation

The goal of texture representation or texture feature extraction is to transform the input texture image into a feature vector that describes the properties of a texture, facilitating subsequent tasks such as texture classification, as illustrated in Fig. 4. Since texture is a spatial phenomenon, texture representation cannot be based on a single pixel, and generally requires the analysis of patterns over local pixel neighborhoods. Therefore, a texture image is first transformed to a pool of local features, which are then aggregated into a global representation for an entire image or region. Since the properties of texture are usually translationally invariant, most texture representations are based on an orderless aggregation of local texture features, such as a sum or max operation.

Early in 1981, Julesz [93] introduced “textons”, which refer to basic image features such as elongated blobs, bars, crosses, and...
terminators, as the elementary units of preattentive human texture perception. However Julesz’s texton studies were limited by their exclusive focus on artificial texture patterns rather than natural textures. In addition, Julesz did not provide a rigorous definition for textons. Subsequently, texton theory fell into disfavor as a model of texture discrimination until the influential work by Leung and Malik [114] who revisited textons and gave an operational definition of a texton as a cluster center in filter response space. This not only enabled textons to be generated automatically from an image, but also opened up the possibility of learning a universal texton dictionary for all images. Texture images can be statistically represented as histograms over a texton dictionary, referred to as the Bag of Textons (BoT) approach. Although BoT was initially developed in the context of texture recognition [114, 143], it was introduced / generalized to image retrieval [216] and classification [42], where it was referred to as Bag of Features (BoF) or, more commonly, Bag of Words (BoW). The research community has since witnessed the prominence of the BoW model for over a decade during which many improvements were proposed.

3.1 The BoW Pipeline

The BoW pipeline is sketched in Fig. 5, consisting of the following basic steps:

1. Local Patch Extraction. For a given image, a pool of $N$ image patches is extracted over a sparse set of points of interest [110, 263], over a fixed grid [101, 150, 207], or densely at each pixel position [163, 236, 237].

2. Local Patch Representation. Given the extracted $N$ patches, local texture descriptors are applied to obtain a set or pool of texture features of $D$ dimension. We denote the local features of $N$ patches in an image as $\{x_i\}_{i=1}^N$, $x_i \in \mathbb{R}^D$. Ideally, local descriptors should be distinctive and at the same time robust to a variety of possible image transformations, such as scale, rotation, blur, illumination, and viewpoint changes. High quality local texture descriptors play a critical role in the BoW pipeline.

3. Codebook Generation. The objective of this step is to generate a codebook (i.e., a texton dictionary) with $K$ codewords $\{w_i\}_{i=1}^K$, $w_i \in \mathbb{R}^D$ based on training data. The codewords may be learned (e.g., by $k$-means [109, 236]) or in a predefined way (such as LBP [163]). The size and nature of the codebook affects the representation followed and thus the discrimination power. The key here is how to generate a compact and discriminative codebook so as to enable accurate and efficient classification.

4. Feature Encoding. Given the generated codebook and the extracted local texture features $\{x_i\}$ from an image, feature encoding represents each local feature $x_i$ with the codebook, usually by mapping each $x_i$ to one or a number of codewords, resulting in a feature coding vector $v_i$ (e.g., $v_i \in \mathbb{R}^K$). Of all the steps in the BoW pipeline, feature encoding is a core component which links local representation and feature pooling, greatly influencing texture classification in terms of both accuracy and speed. Thus, many studies have focused on developing powerful feature encoding, such as vector quantization / $k$-means, sparse coding [139, 140, 181], Locality constrained Linear Coding (LLC) [240], Vector of Locally Aggregated Descriptors (VLAD) [91], and Fisher Vector (FV) [36, 179, 203].

5. Feature Pooling. A global feature representation $y$ is produced by using a feature pooling strategy to aggregate the coded feature vectors $\{v_i\}$. Classical pooling methods include average pooling, max pooling, and Spatial Pyramid Pooling (SPP) [111, 223].

6. Feature Classification. The global feature is used as the basis for classification, for which many approaches are possible [90, 242]: Nearest Neighbor Classifier (NNC), Support Vector Machines (SVM), neural networks, and random forests. SVM is one of the most widely used classifiers for the BoW based representation.

The remainder of this section will introduce the methods in each component, as summarized in Table 1.

3.2 Local Texture Descriptors

All local texture descriptors aim to provide local representations invariant to contrast, rotation, scale, and possibly other criteria. The primary categorization is whether the descriptor is applied densely, at every pixel, as opposed to sparsely, only at certain locations of interest.

3.2.1 Sparse Texture Descriptors

To develop a sparse texture descriptor, a region of interest detector must be designed which is able to reliably detect a sparse set of regions, reliably and stably, under various imaging conditions. Typically, the detected regions undergo a geometric normalization, after which local descriptors are applied to encode the image content. A series of region detectors and local descriptors has been proposed, with excellent surveys [154, 155, 226]. The sparse approach was introduced to texture recognition by Lazebnik et al. [109, 110] and followed by Zhang et al. [263].

In [110] two types of complementary region detectors, the Harris affine detector of Mikolajczyk and Schmid [153] and the Laplacian blob detector of Gårding and Lindeberg [64], were used to detect affine covariant regions, meaning that the region content is affine invariant. Each detected region can be thought of as a texture element having a characteristic elliptic shape and a distinctive appearance pattern. In order to achieve affine invariance, each elliptical region was normalized and then two rotation invariant descriptors, the spin image (SPIN) and the Rotation Invariant Feature Transform (RIFT) descriptor, were applied. As a result, for each texture image four feature channels were extracted (two detectors \(\times\) two descriptors), and for each feature channel $k$ means clustering is performed to form its signature. The Earth Mover’s Distance (EMD) [200] was used for measuring the similarity between image signatures and NNC was used for classification. The Harris affine regions and Laplacian blobs in combination with SPIN and RIFT descriptors (i.e., the (H+L)(S+R) method) have demonstrated good performance [180, 242, 243] in classifying textures with significant affine variations, evidenced by the classification rate 96.0% on UIUC with a NNC classifier.

Following Lazebnik et al. [110], Zhang et al. [263] presented an evaluation of multiple region detector types, levels of geometric
invariance, multiple local texture descriptors, and SVM classifier with kernels based on two effective measures for comparing distributions (signatures and EMD distance vs. standard BoW and the Chi Square distance) for texture and object recognition. Regarding local description, Zhang et al. [263] also used the SIFT descriptor in addition to SPIN and RIFT. With SVM classification, Zhang et al. [263] showed significant performance improvement over that of Lazebnik et al. [110], and reported classification rates of 95.3% and 98.7% on CURET and UIUC respectively. They recommended that practical texture recognition should seek to incorporate multiple types of complementary features, but with local invariance properties not exceeding those absolutely required for a given application. Other local region detectors have also been used for texture description, such as the Scale Descriptors which measure the scales of salient textons [95].

### 3.2.2 Dense Texture Descriptors

The number of features derived from a sparse set of interesting points is much smaller than the total number of image pixels, resulting a compact feature space. However, the sparse approach can be inappropriate for many texture classification tasks:

- Interest point detectors typically produce a sparse output and could miss important texture elements.
- A sparse output in a small image might not produce sufficient regions for robust statistical characterization.
- There are issues regarding the repeatability of the detectors, the stability of the selected regions and the instability of orientation estimation [155].

As a result, extracting local texture features densely at each pixel is the more popular representation, the subject of the following discussion.

1. **Gabor Filters** are one of the most popular texture descriptors, motivated by their relation to models of early visual systems of mammals as well as their joint optimum resolution in time and frequency [89, 113, 148]. As illustrated in Fig. 6, Gabor filters can be considered as orientation and scale tunable edge and bar detectors. The Gabor wavelets are generated by appropriate rotations and dilations from the following product of an elliptical Gaussian and a complex plane wave:

\[
\phi(x, y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left[-\left(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}\right)\right]\exp(j2\pi\omega),
\]

whose Fourier transform is

\[
\hat{\phi}(u, v) = \exp\left[-\left(\frac{(u - \omega)^2}{2\sigma_x^2} + \frac{v^2}{2\sigma_y^2}\right)\right],
\]

where \(\omega\) is the radial center frequency of the filter in the frequency domain, \(\sigma_x\) and \(\sigma_y\) are the standard deviations of the elliptical Gaussian along \(x\) and \(y\).

Thus, a Gabor filter bank is defined by its parameters including frequencies, orientations and the parameters of the Gaussian envelope. In the literature, different parameter settings have been suggested, and filter banks created by these parameter settings work well in general. Details for the derivation of Gabor wavelets and parameter selection can be found in [113, 148, 180]. Invariant Gabor representations can be accessed in [78]. According to the experimental study in [97, 263], Gabor features [148] fail to meet the expected level of performance in the presence of rotation, affine and scale variations. However, Gabor filters encode structural features from multiple orientations and over a broader range of scales. It has been shown [97] that for large datasets, under varying illumination conditions, Gabor filters can serve as a preprocessing method and combine with LBP [163] to obtain texture features with reasonable robustness [185, 264].

2. **Filters by Leung and Malik (LM Filters)** [114, 143] pioneered the problem of classifying textures under varying viewpoint and illumination. The LM filters used for local texture feature extraction are illustrated in Fig. 8. In particular, they marked a milestone by giving an operational definition of textons: the cluster centers of the filter response vectors. Their work has been widely followed by other researchers [42, 110, 210, 216, 236, 237]. To
Approach Highlights

Predefined No codebook learning step; Computationally efficient.

LM Filters [114] First to propose Bag of Texton (BoT) model (*i.e.* the BoW model)

Schmid Filters Gabor like filters; Rotation invariant.

MR8 [236] Rotationally invariant filters and low-dimensional filter response space.

Patch Intensity [237] Challenge the dominant role of filter descriptors and propose image raw intensity feature.

LBP [163] Fast binary features with gray scale invariance; Predefined codebook.

Random Projection [125] First to introduce compressive sensing and random projection into texture classification.

Sort Random Projection [126] Efficient and effective approach for random projection to achieve rotation invariance.

Basic Image Features (BIFs) [40] Introduce BIFs of Griffin and Lillicomb into texture classification; Predefined codebook.

Weber Local Descriptor (WLD) [40] A descriptor based on Weber’s Law.

No codebook learning step; Computationally efficient.

Joint optimum resolution in time and frequency; Multiscale and multiorientation analysis.

Challenge the dominant role of filter descriptors and propose image raw intensity feature.

Fast binary features with gray scale invariance; Predefined codebook.

First to introduce compressive sensing and random projection into texture classification.

Efficient and effective approach for random projection to achieve rotation invariance.

Introduce BIFs of Griffin and Lillicomb into texture classification; Predefined codebook.

A descriptor based on Weber’s Law.

No codebook learning step; Computationally efficient.

Most commonly used method; Cannot capture overlapping distributions in the feature space.

Considers both cluster centers and covariances which describe the spreads of clusters.

Sparse representation based; Minimize reconstruction error of data; Computationally expensive.

Require a large codebook (usually learned by k-means); Usually combine with nonlinear SVM.

Quantize each feature to nearest codeword; Fast to compute; Codes are sparse and high dimensional.

Assigns each feature to multiple codewords; Does not minimize reconstruction error.

Require a small codebook; Very high dimension; Combines with efficient linear SVM.

GMM-based; Encodes higher order statistics; Efficient to compute.

Uses signed square rooting and $L_2$ normalization; State of the art performance in texture classification.

A simplified version of FV.

Enforce sparse representation; Explores the manifold structure of data; Minimize reconstruction error.

Leverage that fact that natural images are sparse; Optimization is computationally expensive.

Local smooth sparsity; Fast computation through approximated LLC.

The most widely used pooling scheme in texture representation.

Usually used in combination with sparse coding and LLC.

Preserving more spatial information; Higher feature dimensionality.

Simple and elegant nonparametric classifier; Popular in texture classification.

Usually in combination with Chi Square for BoW based representation.

Suitable for high-dimensional feature representation like FV and VLAD.

Invariant under the bi-Lipschitz mapping.

Fig. 7 Illustration of the rotationally invariant Gabor-like Schmid filters used in [204]. The parameter $(\sigma, \beta)$ pair takes values (2,1), (4,1), (4,2), (6,1), (6,2), (6,3), (8,1), (8,2), (8,3), (10,1), (10,2), (10,3) and (10,4).

Fig. 6 Illustration of the Gabor wavelets used in [148].

\[
\phi(x, y) = \exp \left[ -\frac{x^2 + y^2}{2\sigma^2} \right] \cos \left( \frac{\pi \beta \sqrt{x^2 + y^2}}{\sigma} \right),
\]

where $\beta$ is the number of cycles of the harmonic function within the Gaussian envelope of the filter. The filters are shown in Fig. 7; as can be seen, all of the filters have rotational symmetry. The rotation-invariant S Filters were shown to outperform the rotation-invariant LM Filters in classifying the CUREt textures [236], indicating that rotational invariance is necessary in practical applications.
results [236] showed that MR8 outperformed the LM Filters and S Filters, indicating that detecting better features and clustering in a lower dimensional feature space can be advantageous. The best results for MR8 are 97.4% obtained with a dictionary of 2440 textons and a Nearest Neighbor Classifier (NNC) [236]. Later, Hayman et al. [82] showed that SVM could further enhance the texture classification performance of MR8 features, giving a 98.5% classification rate for the same setup used for texton representation.

(5) Patch Descriptors of Varma and Zisserman [237] challenged the dominant role of the filter banks [152, 190] in texture analysis, and instead developed a simple Patch Descriptor, keeping the raw pixel intensities of a square neighborhood to form a feature vector, as illustrated in Fig. 10. By replacing the filter responses such as LM Filters [190], S Filters [204] and MR8 [236] with the Patch Descriptor in texture classification, Varma and Zisserman [237] observed very good classification performance using extremely compact neighborhoods (3 × 3), and that for any fixed size of neighborhood the Patch Descriptor leads to superior classification compared to filter banks with the same support.

Two variants of the Patch Descriptor, the Neighborhood Descriptor and the MRF Descriptor, were developed. For the Neighborhood Descriptor, the central pixel is discarded and only the neighborhood vector is used for texton representation. Instead of ignoring the central pixel, the MRF Descriptor explicitly models the joint distribution of the central pixels and its neighbors. The best result 98.0% is given by the MRF Descriptor using a 7 × 7 neighborhood with 2440 textons and 90 bins and a NNC classifier. Note that the dimensionality of this MRF representation is very high: 2440 × 90. A clear limitation of the Patch, Neighborhood and MRF Descriptors is sensitivity to nearly any change (brightness, rotation, affine etc.). Varma and Zisserman [237] adopted the method of finding the dominant orientation of a patch and measuring the neighborhood relative to this orientation to achieve rotation invariance, and reported a 97.8% classification rate on the UIUC dataset. It is worth mentioning that finding the dominant orientation for each patch is computationally expensive.

(6) Random Projection (RP) and Sorted Random Projection (SRP) features of Liu and Fieguth [125] were inspired by theories of sparse representation and compressed sensing [27, 52]. Taking advantage of the sparse nature of textured images, a small set of random features is extracted from local image patches by projecting the local patch feature vectors to a lower dimensional feature subspace. The random projection is a fixed, distance-preserving embedding capable of alleviating the curse of dimensionality [11, 69]. The random features are embedded into BoW to perform texture classification. It has been shown that the performance of RP features is superior to that of the Patch Descriptor with equivalent neighborhoods [125]; a clear indication that the RP matrix preserves the salient information contained in the local patch and that performing classification in a lower feature space is advantageous.

The best result 98.5% is achieved using a 17 × 17 neighborhood with 2440 textons and a NNC classifier.

Like the Patch Descriptors, the RP features remain sensitive to image rotation. To further improve robustness, Liu et al. [128, 126] proposed sorting the RP features, as illustrated in Fig. 11, whereby rings of pixel values are sorted, without any reference orientation, ensuring rotation invariance. Two kinds of local features are used, one based on raw intensities and the other on gradients (radial dif-
### Local SRP Feature Extraction

![Sorted Pixel Values](image1)

![Sorted Pixel Differences](image2)

**Fig. 11** An illustration of SRP descriptor: extracting SRP features on an example local image patch of size $7 \times 7$. (a) Sorting pixel intensities; (b,c) Sorting pixel differences.

![Circular](image3)

![Radial-DIFF](image4)

![Angular-DIFF](image5)

**Fig. 12** A circular neighborhood used to derive an LBP code: a central pixel $x_c$ and its $p$ circularly and evenly spaced neighbors on a circle of radius $r$.

In this joint distribution is conveyed by the joint distribution of differences:

$$g(x_0 - x_c, x_1 - x_c, \ldots, x_p - x_c).$$

(2)

The size of the joint histogram was greatly minimized by keeping only the **sign** of each difference, as illustrated in Fig. 12.

A certain degree of rotation invariance is achieved by cyclic shifts of the LBPs, i.e., grouping together those LBPs that are actually rotated versions of the same underlying pattern. Since the dimensionality of the representation (which grows exponentially with $p$) is still high, Ojala et al. [163] introduced a uniformity measure to identify $p(p-1)/2$ uniform LBPs and classified all remaining nonuniform LBPs under a single group. By changing parameters $p$ and $r$, we can derive LBP for any quantization of the angular space and for any spatial resolution, such that multiscale analysis can be accomplished by combining multiple operators of varying $r$. The most prominent advantages of LBP are its invariance to monotonic gray scale change, very low computational complexity, and ease of implementation.

Since [163], LBP started to receive increasing attention in computer vision and pattern recognition, especially texture and facial analysis, with the LBP milestones presented in Fig. 13. As Gabor filters and LBP provide complementary information (LBP captures small and fine details, Gabor filters encode appearance information over a broader range of scales), Zhang et al. [264] proposed Local Gabor Binary Pattern (LGBP) by extracting LBP features from images filtered by Gabor filters of different scales and orientations, to enhance the representation power, followed by subsequent Gabor-LBP approaches [86, 132, 185]. Additional important LBP variants include LBP-TOP, proposed by Zhao and Pietikäinen [265], a milestone in using LBP for dynamic texture analysis; the Local Ternary Pattern (LTP) of Tan and Triggs [222], introducing a pair of thresholds and a split coding scheme which allows for encoding pixel similarity; the Local Phase Quantization (LPQ) by Ojansivu et al. [164, 165] quantizing the Fourier transform phase in local neighborhoods which is, by design, tolerant to most common types of image blurs; the Completed LBP (CLBP) of Guo et al. [76], encoding not only the signs but also the magnitudes of local differences; and the Median Robust Extended LBP (MRELBP) of Liu et al. [131] which enjoys high distinctiveness, low computational complexity, and strong robustness to image rotation and noise.

LBP has also led to compact and efficient binary feature descriptors designed for image matching, with noticeable in-
Fig. 14 Illustration of the calculation of BIF features.

In the image is filtered by the DtG filters, and then labeled as the maximizing class. A simple six dimensional BIF histogram can be used as a global texture representation, however the histogram over these six categories produces too coarse a representation, therefore others (e.g., Crosier and Griffin [40]) have performed multiscale analysis and calculated joint histograms over multiple scales. Multiscale BIF features achieved very good classification performance on CUREt (98.6%), UIUC (98.8%) and KTHTIPS (98.5%) [40], with a NNC classifier.

(9) Weber Law Descriptor (WLD) [33] is based on the fact that human perception of a pattern depends not only on the change of a stimulus but also on the original intensity of the stimulus. The WLD consists of two components: differential excitation and orientation. For a small patch of size $3 \times 3$, shown in Fig. 15, the differential excitation is the relative intensity ratio

$$\zeta(x_c) = \arctan\left(\frac{\sum_{i=0}^{7}(x_i - x_c)}{x_c}\right)$$

and the orientation component is derived from the local gradient orientation

$$\theta(x_c) = \arctan\left(\frac{x_7 - x_3}{x_5 - x_1}\right).$$

Both $\zeta$ and $\theta$ are quantified into a 2D histogram, offering a global representation. Clearly the use of multiple neighborhood sizes supports a multiscale generalization. Though computationally efficient, WLD features fail to meet the expected level of performance for texture recognition.

3.2.3 Fractal Based Descriptors

Fractal Based Descriptors present a mathematically well founded alternative to dealing with scale [146], however they have not become popular as texture features due to their lack of discriminative power [235]. Recently, inspired by the BoW approach, researchers revisited the fractal method and proposed the MultiFractal Spectrum (MFS) method [253, 252, 254], invariant to viewpoint changes, nonrigid deformations and local affine illumination changes.

The basic MFS method was proposed in [253], where MFS was first defined for simple image features, such as intensity, gradient and Laplacian of Gaussian (LoG). A texture image is first transformed into $n$ feature maps such as intensity, gradient or LoG filter features. Each map is clustered into $k$ clusters (i.e. $k$ codewords) via $k$means. Then, a codeword label map is obtained and is decomposed into $k$ binary feature maps: those pixels assigned to codeword $i$ are labeled with 1 and the remainder as 0. For each binary feature map, the box counting algorithm [254] is used to estimate a fractal dimension feature. Thus, a total of $k$ fractal dimension features are computed for each feature map, forming a $kD$
feature vector (referred to as a fractal spectrum) as the global representation of the image. Finally, for the \( n \) different feature maps, \( n \) fractal spectrum feature vectors are concatenated as the MFS feature. The MFS representation demonstrated invariance to a number of geometrical changes such as viewpoint changes, nonrigid surface changes and reasonable robustness to illumination changes. However, since it is based on simple features (intensities and gradients) and has very low dimension, it has limited discriminability, and gives classification rates 92.3\% and 93.9\% on datasets UIUC and UMD respectively.

Later MFS was improved by generalizing the simple image intensity and gradient features with SIFT [252], wavelets [254], and LBP [188]. For instance, the Wavelet based MFS (WMFS) features archived significantly improved classification performance on UIUC (98.6\%) and UMD (98.7\%). The downside of the MFS approach is that it requires high resolution images to obtain sufficiently stable features.

### 3.3 Codebook Generation

Texture characterization requires the analysis of spatially repeating patterns, which suffice to characterize textures and the pursuit of which has had important implications in a series of practical problems, such as dimensionality reduction, variable decoupling, and biological modelling [168, 272]. The extracted set of local texture features is versatile, and yet overly redundant [114]. It can therefore be expected that a set of prototype features (i.e. codewords or textons) must exist which can be used to create global representations of textures in natural images [114, 166, 272], in a similar way as in speech and language (such as words, phrases and sentences).

There exist a variety of methods for codebook generation. Certain approaches, such as LBP [163] and BIF [40], which we have already discussed, use predefined codebooks, therefore entirely bypassing the codebook learning step.

For approaches requiring a learned codebook, \( k \)means clustering [110, 114, 125, 237, 263] and Gaussian Mixture Models (GMM) [34, 36, 107, 91, 179, 209] are the most popular and successful methods. GMM modeling considers both cluster centers and covariances, which describe the location and spread/shape of clusters, whereas \( k \)means clustering cannot capture overlapping distributions in the feature space as it considers only distances to cluster centers, although generalizations to \( k \)means with multiple prototypes per cluster can allow this limitation to be relaxed. The GMM and \( k \)means methods learn a codebook in an unsupervised manner, but some recent approaches focus on building more discriminative ones [257, 246].

In addition, another significant research thread is reconstruction based codebook learning [1, 181, 217, 240], under the assumption that natural images admit a sparse decomposition in some redundant basis (i.e., dictionary or codebook). These methods focus on learning nonparametric redundant dictionaries that facilitate a sparse representation of the data and minimize the reconstruction error of the data. Because discrimination is the primary goal of texture classification, researchers have proposed to construct discriminative dictionaries that explicitly incorporate category specific information [139, 140].

Since the codebook is used as the basis for encoding feature vectors, codebook generation is often interleaved with feature encoding, described next.

### 3.4 Feature Encoding

As illustrated in Fig. 4, a given image is transformed into a pool of local texture features, from which a global image representation is derived by feature encoding with the generated codebook. In the field of texture classification, we group commonly-used encoding strategies into three major categories:

- **Voting based methods.** The most intuitive way to quantize a local feature is to assign it to its nearest codeword in the codebook, also referred to as hard voting [114, 236]. A histogram of the quantized local descriptors can be computed by counting the number of local features assigned to each codeword; this histogram constitutes the baseline BoW representation (as illustrated in Fig. 16 (a)) upon which other methods can improve. Formally, it starts by learning a codebook \( \{w_k\}_{k=1}^K \), usually by \( k \)means clustering. Given a set of local texture descriptors \( \{x_i\}_{i=1}^N \) extracted from an image, the encoding representation of some descriptor \( x \) via hard voting is

\[
v(i) = \begin{cases} 
1, & \text{if } \hat{i} = \arg\min_j (\|x - w_j\|) \\
0, & \text{otherwise.} 
\end{cases}
\]  

(3)

The histogram of the set of local descriptors is to aggregate all encoding vectors \( \{v_i\}_{i=1}^N \) via sum pooling. Hard voting overlooks codeword uncertainty, and may label image features by nonrepresentative codewords. In an improvement to this hard voting scheme, soft voting [2, 196, 258, 232, 233] employs several nearest codewords to encode each local feature in a soft manner, such that the weight of each assigned codeword is an inverse function of the distance from the feature, for some kernel definition of distance. Voting based methods yield a histogram representation of dimensionality \( K \), the number of bins in the histogram.

![Fig. 16 Contrasting the ideas of BoW, VLAD and FV. (a) BoW: Counting the number of local features assigned to each codeword. It encodes the zero order statistics of the distribution of local descriptors. (b) VLAD: Accumulating the differences of local features assigned to each codeword. (c) FV: The Fisher vector extends the BoW by encoding higher order statistics (first and second order), retaining information about the fitting error of the best fit.](image)
Fisher Vector based methods. By counting the number of occurrences of codewords, the standard BoW histogram representation encodes the zeroth-order statistics of the distribution of descriptors, which is only a rough approximation of the probability density distribution of the local features. The Fisher vector extends the histogram approach by encoding additional information about the distribution of the local descriptors. Based on the original FV encoding [178], improved versions were proposed [37, 179] such as the Improved FV (IFV) [179] and VLAD [91]. We briefly describe IFV [179] here, since to the best of our knowledge it achieves the best performance in texture classification [34, 35, 36, 209]. Theory and practical issues regarding FV encoding can be found in [203].

IFV encoding learns a soft codebook with GMM, as shown in Fig. 16 (c). An IFV encoding of a local feature is computed by assigning it to each codeword, in turn, and computing the gradient of the soft assignment with respect to the GMM parameters\(^2\). The IFV encoding dimensionality is \(2DK\), where \(D\) is the dimensionality of the feature space and \(K\) is the number of Gaussian mixtures. BoW can be considered a special case of FV in the case where the gradient computation is restricted to the mixture weight parameters of the GMM. Unlike BoW, which requires a large codebook size, FV can be computed from a much smaller codebook (typically 64 or 256) and therefore at a lower computational cost at the codebook learning step. On the other hand, the resulting dimension of the FV encoding vector (e.g. tens of thousands) is usually significantly higher than BoW (thousands), which makes it unsuitable for nonlinear classifiers, however it offers good performance even with simple linear classifiers.

The VLAD encoding scheme proposed by Jégou et al. [91] can be thought of as a simplified version of FV, in that it typically uses kmeans, rather than GMM, and records only first-order statistics rather than second order. In particular, it records the residuals (the difference between the local features and the codewords), as shown in Fig. 16 (b).

Reconstruction based methods. Reconstruction based methods aim to obtain an information-preserving encoding vector that allows for the reconstruction of a local feature with a small number of codewords. Typical methods include sparse coding and Local constraint Linear Coding (LLC), which are contrasted in Fig. 17. Sparse coding was initially proposed [167] to model natural image statistics, then to texture classification [45, 139, 140, 181, 217] and later to other problems such as image classification [256] and face recognition [247].

In sparse coding, a local feature \(x\) can be well approximated by a sparse decomposition \(x \approx \text{W}v\) over the learned codebook \(\text{W} = [w_1, w_2, \ldots, w_K]\), by leveraging the sparse nature of the underlying image [167]. A sparse encoding can be solved as

\[
\text{argmin}_v \|x - \text{W}v\|_2^2 \quad \text{s.t.} \quad \|v\|_0 \leq s. \tag{4}
\]

where \(s\) is a small integer denoting the sparsity level, limiting the number of nonzero entries in \(v\), measured as \(\|v\|_0\). Learning a redundant codebook that facilitate a sparse representation of the local features is important in sparse coding [11]. Methods in [139, 140, 181, 217] are based on learning \(C\) class-specific codebooks, one for each texture class and approximating each local feature using a constant sparsity \(s\). The \(C\) different codebooks provides \(C\) different reconstruction errors, which can then be used as classification features. In [181, 217], the class specific codebooks were optimized for reconstruction, but significant improvements have been shown by optimizing for discriminative power instead [45, 139, 140], an approach which is, however, associated with high computational cost, especially when the number of texture classes \(C\) is large.

Locality constrained linear coding (LLC) [240] projects each local descriptor \(x\) down to the local linear subspace spanned by \(q\) codewords in the codebook of size \(K\) closest to it (in Euclidean distance), resulting in a \(K\) dimensional encoding vector whose entries are all zero except for the indices of the \(q\) codewords closest to \(x\). The projection of \(x\) down to the span of its \(q\) closest codewords is solved via

\[
\text{argmin}_v \|x - \text{W}v\|_2^2 + \lambda \sum_{k=1}^{K} \left( \frac{v(i) \exp \frac{\|x - w_i\|_2^2}{\sigma}}{\sigma} \right)^2 \quad \text{s.t.} \sum_{k=1}^{K} v(i) = 1,
\]

where \(\lambda\) is a small regularization constant and \(\sigma\) adjusts the weight decay speed.

In summary, reconstruction based coding has received significant attention since sparse coding was applied for visual classification [139, 140, 181, 217, 240]. A theoretical study for the success of sparse coding over vector quantization can be found in [38].

3.5 Feature Pooling and Classification

The goal of feature pooling [19] is to integrate or combine the coded feature vectors \(\{v_1, \ldots, v_j\} \in \mathbb{R}^d\) of a given image into a final compact global representation \(y\), which is more robust to image transformations and noise. Commonly used pooling methods include sum pooling, average pooling and max pooling [114, 237, 240]. Boureau et al. [19] presented a theoretical analysis of average pooling and max pooling, and showed that max pooling may be well suited to sparse features. The authors also proposed softer max pooling methods by using a smoother estimate of the expected max-pooled feature and demonstrated improved performance. Another noticeable pooling method is the mix-order max pooling method which considers the information of visual word occurrence frequency [127].

Specifically, let \(V = [v_1, \ldots, v_N] \in \mathbb{R}^{d \times N}\) denote the coded features from \(N\) locations. For \(u\) denoting a row of \(V\), \(u\) is reduced to a single scalar by some operation (sum, average, max), reducing \(V\) to a \(d\)-dimensional feature vector. Realizing that pooling over
4 CNN Based Texture Representation

A large number of CNN-based texture representation methods have been proposed in recent years since the record-breaking image classification result [103] achieved in 2012. A key to the success of CNNs is their ability to leverage large labeled datasets to learn high quality features. Learning CNNs, however, amounts to estimating millions of parameters and requires a very large number of annotated images, an issue which rather constrains the application of CNNs in problems with limited training data. A key discovery, in this regard, is that CNN features pretrained on very large datasets were found to transfer well to many other problems, including texture analysis, with a relatively modest adaptation effort [31, 36, 68, 170, 208]. In general, the current literature on texture classification includes examples of both employing pretrained generic CNN models or performing finetuning for specific texture classification tasks.

In this survey we will classify CNN based texture representation methods into three categories, and which form the basis of the following three sections:

- using pretrained generic CNN models,
- using finetuned CNN models, and
- using handcrafted deep convolutional networks.

These representations have had a widespread influence in image understanding; representative examples of each of these are given in Table 2.

4.1 Using Pretrained Generic CNN Models

Given the behavior of CNN transfer, the success of pretrained CNN models lies in the feature extraction and encoding steps. Similar to Section 3, we will describe first some commonly used networks for pretraining and then the feature extraction process.

(1) Popular Generic CNN Models

These networks, AlexNet was proposed the earliest, and in general the others are deeper and more complex. A full review of these networks is beyond the scope of this paper, and we refer readers to the original papers [84, 87, 103, 214, 220] and to excellent surveys [15, 31, 75, 112, 133] for additional details. Briefly, as shown in Fig. 18 (b), a typical CNN repeatedly applies the following three operations:

1. Convolution with a number of linear filters,
2. Nonlinearities, such as sigmoid or rectification,
3. Local pooling or subsampling.

These three operations are highly related to traditional filter bank methods widely used in texture analysis [190], as shown in Fig. 18...
with the key differences that the CNN filters are learned directly from data rather than handcrafted, and that CNNs have a hierarchical architecture learning increasingly abstract levels of representation. These three operations are also closely related to the RP approach (Fig. 18 (c)) and the LBP (Fig. 18 (d)).

Several large-scale image datasets are usually used for CNN pretraining. Among them the commonly used ImageNet dataset, with 1000 classes and 1.2 million images [201], and the scene-centric MITPlaces dataset [267, 268].

Comprehensive evaluations of the feature transfer effect of CNNs for the purpose of texture classification have been conducted in [34, 35, 36, 159], with the following critical insights. During model transfer, features extracted from different layers exhibit different classification performance. Experiments confirm that the fully-connected layers of the CNN, whose role is primarily that of classification, tend to exhibit relatively worse generalization ability and transferability, and therefore would need retraining or finetuning on the transfer target. In contrast the convolutional layers, which act more as feature extractors, with coarser convolutional layers acting as progressively more abstract features, generally transfer well. That is, the convolutional descriptors are substantially less committed to a specific dataset than the fully connected descriptors. As a result, the source training set is relevant to classification accuracy on different datasets, and the similarity of the source and target plays a critical role when using a pretrained CNN model [14]. Finally, from [35, 36, 159] it was found that deeper models transfer better, and that the deepest convolutional descriptors give the best performance, superior to the fully-connected descriptors, when proper encoding techniques are employed (such as FVCNN+—CNN features with Fisher Vector encoder).

(3) Feature Encoding and Pooling: A set of features extracted from convolutional or fully connected layers resembles a set of texture features as described in Section 3.2, so the traditional feature encoding methods discussed in Section 3.4 can be directly employed.
In [36], Cimpoi et al. evaluated several encoders, i.e. standard BoW [114], LLC [240], VLAD [91] and IFV [179] (reviewed in Section 3.4), for CNN features, and showed that the best performance is achieved by IFV. It has been reported that VGGVD+IFV with a linear SVM classifier produced consistently near perfect classification performance on several texture datasets; KTHTIPS (99.8%), UIUC (99.9%), UMD (99.9%) and ALOT (99.5%), as summarized in Table 4. In addition, it obtained significant improvement on very challenging datasets like KTHTIPS2b (81.8%), FMD (79.8%) and DTD (72.3%). However, it only achieved 80.0% and 82.3% on Outex_TC10 and Outex TC12 respectively, which are significantly worse than the near perfect performance of MRELBP on these two datasets [132]; a clear indicator that DCNN based features require large amount of training samples and that they lack local invariance. Song et al. [218] proposed a neural network to transform the FCVNN descriptors into a lower dimensional representation. As shown in Fig. 20, locally transferred FCVNN (LFVCNN) descriptors are obtained by passing the $2K D$ dimensional FCVNN descriptors of images through a multilayer neural network consisting of fully connected, $l_2$ normalization layers, and ReLU layers. LFVCNN achieved state of the art results on KTHTIPS2b (82.6%), FMD (82.1%) and DTD (73.8%), as shown in Table 4.

Recently, Gatys et al. [65] showed that the Gram matrix representations extracted from various layers of VGGNet [214] can be inverted for texture synthesis. The work of Gatys et al. [65] ignited a renewed interest in texture synthesis [229]. Notably, the Gram matrix representation used in their approach is identical to the bilinear pooling of CNN features of Lin et al. [122], which were proved to be good for texture recognition in [120]. Like the traditional encoders introduced in Section 3.4, the bilinear feature pooling is an orderless representation of the input image and hence is suitable for modeling textures. The Bilinear CNN (BCNN) descriptors are obtained by computing the outer product of each feature $x^i$ with itself, reordered into feature vectors, and subsequently pooled via sum to obtain the final global representation. The dimension of the bilinear descriptor is $(D^i)^2$, which is very high (e.g. 512²). It was shown in [120, 123] that the texture classification performance of BCNN and FCVNN was virtually identical, indicating that bilinear pooling is as good as the Fisher vector pooling for texture recognition. It was also found that the BCNN descriptor of the last convolutional layer performed the best, in agreement with [36].

### 4.2 Using Finetuned CNN Models:

Pretrained CNN models, discussed in Section 4.1, have achieved impressive performance in texture recognition, however training in these methods is a multistage pipeline that involves feature extraction, codebook generation, feature encoding and classifier training. Consequently, these methods cannot take advantage of utilizing the full capability of neural networks in representation learning. Generally finetuning CNN models on task-specific training datasets (or learning from scratch if large-scale task-specific datasets are available) is expected to improve on already strong performance achieved by pretrained CNN models [31, 68]. When using a fine-tuned CNN model, the global image representation is usually generated in an end-to-end manner; that is, the network will produce a final visual representation without additional explicit encoding or pooling steps, as illustrated in Fig. 5. When finetuning a CNN, the last fully connected layer is modified to have $B$ nodes corresponding to the number of classes in the target dataset. The nature of the datasets used in finetuning is important to learning discriminative CNN features. The pretrained CNN model is capable of discriminating images of different objects or scene classes, but may be less effective in discerning the difference between different textures (material types) since an image in ImageNet may contain different types of textures (materials). The size of the dataset used in finetuning matters as well, since too small a dataset may be inadequate for complete learning.

To the best of our knowledge, the behaviour of a finetuned large-scale CNN like VGGNet [214] or training it from scratch using a texture dataset have not been fully explored, almost certainly due to the fact that a large texture dataset on the scale of ImageNet [201] or MITPlaces [267] does not exist. Most existing texture datasets are small, as discussed later in Section 6, and according to [9, 120] finetuning a VGGNet [214] or AlexNet [103] on existing texture datasets leads to negligible performance improvement. As shown in Fig. 19 (a), for a typical CNN like VGGNet [214], the output of the last convolutional layer is reshaped into a single feature vector (spatially sensitive) and fed into fully connected layers (i.e., order sensitive pooling). The global spatial information is necessary for analyzing the global shapes of objects, however it has been realized [9, 36, 65, 120, 261] that it is not of great importance for analyzing textures due to the need for orderless representation. The FCVNN descriptor shows higher recognition performance than FCCNN, even if the pretrained VGGVD model is fine-
tuned on the texture dataset (i.e., the finetuned FCCNN descriptor) [36, 120]. Therefore, an orderless feature pooling from the output of a convolution layer is desirable for end-to-end learning. In addition, orderless pooling does not require an input image to be of a fixed size, motivating a series of innovations in designing novel CNN architectures for texture recognition [9, 10, 46, 123, 261].

A Texture CNN (TCNN) based on AlexNet, as illustrated in Fig. 19 (b), was developed in [9]. It simply utilizes global average pooling to transform a field of descriptor \( X \) at a given convolutional layer \( l \) of a CNN into a \( D \times D \) dimension vector which is connected to a fully connected layer. TCNN has fewer parameters and lower complexity than AlexNet. In addition, Andrearczyk and Whelan [9] proposed to fuse the global average pooled vector of an intermediate convolutional layer and that of the last convolutional layer via concatenation and introduced to later fully connected layers, a combination which resembles the hypercolumn feature developed in [81]. Andrearczyk and Whelan [9] observed that finetuning a network that was pretrained on a texture-centric dataset achieves better results on other texture datasets compared to a network pretrained on an object-centric dataset of the same size, and that the size of the dataset on which the network is pretrained or finetuned predominantly influences the performance of the finetuning. These two observations suggest that a very large texture dataset could bring a significant contribution to CNNs applied to texture analysis.

In BCNN [123], as shown in Fig. 19 (c), Lin et al. proposed to replace the fully connected layers with an orderless bilinear pooling layer, which was discussed in Section 4.1. This method was successfully applied to texture classification in [120] and obtained slightly better results than FVCNN, however the representational power of bilinear features comes at the cost of very high dimensional feature representations, which induce substantial computational burdens and require large amounts of training data, motivating several improvements on BCNN. Gao et al. [63] proposed compact bilinear pooling, as shown in Fig. 19 (d), which utilizes Random Maclaurin Projection or Tensor Sketch Projection to reduce the dimensionality of bilinear representations while still maintaining similar performance to the full BCNN feature [123] with a 90% reduction in the number of learned parameters. To combine the ideas in [9] and [63], Dai et al. [46] proposed an effective fusion network called FASON (First And Second Order information Fusion Network) that combines first and second order information flow, as illustrated in Fig. 19 (e). These two types of features were generated from different convolutional layers and concatenated to form a single feature vector which was connected to a fully connected softmax layer for end to end training. In [100], Kong and Fowlkes proposed to represent the bilinear features as a matrix and applied a low rank bilinear classifier. The resulting classifier can be evaluated without explicitly computing the bilinear feature map which allows for a large reduction in the computational time as well as decreasing the effective number of parameters to be learned.

There are some works attempting to integrate CNN and VLAD or FV pooling in an end to end manner. In [10], a NetVLAD network was proposed by plugging a VLAD-like layer into a CNN network at the last convolutional layer and allows training end to end. The model was initially designed for place recognition, however when applied to texture classification by Song et al. [218] it was found that the classification performance was inferior to FVCNN. Similar to NetVLAD [10], a Deep Texture Encoding Network (DeepTEN) was introduced in [261] by integrating an encoding layer on top of convolutional layers, also generalizing orderless pooling methods such as VLAD and FV in a CNN trained end to end.

4.3 Using Handcrafted Deep Convolutional Networks

In addition to the CNN based methods reviewed in Sections 4.1 and 4.2, some “handcrafted” 3 deep convolutional networks [24, 29] deserve attention. Recall that a standard CNN architecture (as shown in Fig. 18 (b)) consists of multiple trainable building blocks stacked on top of one another followed by a supervised classifier. Each block generally consists of three layers: a convolutional filter bank layer, a nonlinear layer, and a feature pooling layer. Similar to the CNN architecture, Bruna and Mallat [24] proposed a highly influential Scattering convolution Network (ScatNet), as illustrated in Fig. 21.

The key difference from CNN, where the convolutional filters are learned from data, is that the convolutional filters in ScatNet are predetermined — they are simply wavelet filters, such as Gabor or Haar wavelets, and no learning is required. Moreover, the ScatNet usually cannot go as deep as a CNN; Bruna and Mallat [24] suggested two convolutional layers, since the energy of the third layer scattering coefficients is negligible. Specifically, as can be seen in Fig. 21, ScatNet cascade wavelet transform convolutions with modulus nonlinearity and averaging pools. It is shown in [24] that ScatNet computes translation-invariant image representations which are stable to deformations and preserve high frequency information for recognition. As shown in Fig. 21, the average pooled feature vector from each stage is concatenated to form the global feature representation of an image, which is input into a simple PCA classifier for recognition, and which has demonstrated very high performance in texture recognition [24, 212, 213, 211, 132]. It achieved very high classification performance on Outex_TC10 (99.7%), Outex_TC12 (99.1%), KTHTIPS (99.4%), CUReT (99.8%), UIUC (99.4%) and UMD (99.7%) [24, 213, 132], but performed poorly on even challenging datasets like DTD (35.7%). A downside of ScatNet is that the feature extraction stage is very time consuming, although the dimensionality of the global representation feature is relatively low (several hundreds). ScatNet has been extended to achieve rotation and scale invariance [212, 213, 211] and applied to other problems besides texture such as object recognition [173]. Importantly, the mathematical analysis of ScatNet explains important properties of CNN architectures, and it is one of the few works that provides detailed theoretical understanding of CNNs.

Fig. 21 contrasts ScatNet and PCANet, proposed by Chan et al. [29], a very simple convolutional network based on trained PCA filters, instead of predefined Gabor wavelets, and LBP encoding [163] and histogramming for feature pooling. Two simple variations of PCANet, RandNet and LDANet, were also introduced in [29], sharing the same topology as PCANet, but their convolutional

3 Note that “handcrafted” commonly used for traditional features is somewhat imprecise, because many traditional features like Gabor filters are biologically or psychologically inspired.
filters are either random filters as in [125] or learned from Linear Discriminant Analysis (LDA). Compared with ScatNet, feature extraction in PCANet is much faster, but with weaker invariance and texture classification performance [132].

5 Attribute-Based Texture Representation

In recent years, the recognition of texture categories has been extensively studied and has shown substantial progress, partly thanks to the texture representations reviewed in Sections 3 and 4. Despite the rapid progress, particularly with the development of deep learning techniques, we remain far from reaching the goal of comprehensive scene understanding [102], which would support important applications such as detailed image search, question answering, and robotic interactions. Texture attributes are an important component of visual attributes, particularly for objects that are best characterized by a pattern. It can support advanced image search applications, such as more specific queries in image search engines (e.g. a striped skirt, rather than just any skirt). The investigation of texture attributes and detailed semantic texture description offers a significant opportunity to close the semantic gap in texture modeling and to support applications that require fine grained texture description. Nevertheless, there are only several papers [17, 34, 151] investigating the texture attributes thus far, and there is no systematic study yet attempted.

There are three essential issues in studying texture attribute based representation:

1. The identification of a universal texture attribute vocabulary that can describe a wide range of textures;
2. The establishment of a benchmark texture dataset, annotated by semantic attributes;
3. The reliable estimation of texture attributes from images, based on low level texture representations, such as the methods reviewed in Sections 3 and 4.

Tamura et al. [221] proposed a set of six attributes for describing textures: coarseness, contrast, directionality, line-likeness, regularity, and roughness. Amadasun and King [8] refined this idea with the five attributes of coarseness, contrast, business, complexity, and strength. Later, Bhushan et al. [16] studied texture attributes from the perspective of psychology, asking subjects to cluster a collection of 98 texture adjectives according to similarity and identified eleven major clusters.

Recently, inspired by the work in [16, 58, 175, 105], Matthews et al. [151] attempted to enrich texture analysis with semantic attributes. They identified eleven commonly-used texture attributes by selecting a single adjective from each of the eleven clusters

4 Blemished, bumpy, lined, marbled, random, repetitive, speckled, spiralled, webbed, woven, and wrinkled
identified by Bhushan et al. [16]. Then, with the eleven texture attributes, they released a publicly available human-provided labeling of over 300 classes of texture from the Outex database [162]. For each texture image, instead of asking a subject to simply identifying the presence or absence of each texture attribute, Matthews et al. [151] proposed a framework of pairwise comparison, in which a subject was shown two texture images simultaneously and prompted to choose the image exhibiting more of some attribute, motivated by the use of relative attributes [175].

After performing a screening process on the 98 adjectives identified by Bhushan et al. [16], Cimpoi et al. [34] obtained a texture attribute vocabulary of 47 English adjectives and collected a dataset providing 120 example images for each attribute. They furthermore provide a comparison of BoW- and CNN-based texture representation methods for attribute estimation, demonstrating that texture attributes are excellent texture descriptors, transferring between datasets. Bormann et al. [17] introduced a set of seventeen human comprehensible attributes (seven color and ten structural) for color texture characterization. They also collected a new database named Robotics Domain Attributes Database (RDAD) for the indoor service robotics context. They compared five low level texture representation approaches for attribute prediction, and found that not all objects can be described very well with the seventeen attributes. Clearly, which attributes are best suited for a precise description of different object and texture classes deserves further attention.

6 Texture Datasets and Performance

6.1 Texture Datasets

Datasets have played an important role throughout the history of visual recognition research. They have been one of the most important factors for the considerable progress in the field, not only as a common ground for measuring and comparing performance of competing algorithms but also pushing the field towards increasingly complicated and challenging problems. With the rapid development of visual recognition approaches, datasets have become progressively more challenging, evidenced by the fact that the recent large scale ImageNet dataset [201] has enabled breakthroughs in visual recognition research. In the big data era, it becomes urgent to further enrich texture datasets to promote future research. In this section, we discuss existing texture image datasets that have been released and commonly used by the research community for texture classification, as summarized in Table 3.

The Brodatz texture database [22], derived from the Brodatz Album [23], is the earliest, the most widely used and the most famous texture database. It has a relatively large number of classes (111), with each class having only one image. Many texture representation approaches exploit the Brodatz database for evaluations [99, 125, 163, 187, 190, 231], however in most cases the entire database is not utilized, except in some recent studies [67, 110, 132, 182, 263]. The database has been criticized because of the lack of intraclass variations such as scale, rotation, perspective and illumination.

The Vision Texture Database (VisTex) [134, 239] is an early and well-known database. Built by the MIT Multimedia Lab, it has 167 classes of textures, each with only one image. The VisTex textures are imaged under natural lighting conditions, and have extra visual cues such as shadows, lighting, depth, perspective, thus closer in appearance to real-world images. VisTex is often used for texture synthesis or segmentation, but rarely for image-level texture classification.

Since 2000, texture recognition has evolved to classifying real world textures with large intraclass variations due to changes in camera pose and illumination, leading to the development of a number of benchmark texture datasets based on various real-world material instances. Among these, the most famous and widely used is the Columbia-Utrecht Reflectance and Texture (CUREt) dataset [47], with 61 different material textures taken under varying image conditions in a controlled lab environment. The effects of specularities, interreflections, shadowing, and other surface normal variations are evident, as shown in Fig. 3 (a). CUREt is a considerable improvement over Brodatz, where all such effects are absent. Based on the original CUREt, Varma and Zisserman [236] built a subset for texture classification, which became the widely used benchmark to assess classification performance. CUREt has limitations of no significant scale change for most of the textures and limited in-plane rotation. Thus, a discriminative texture feature without rotation invariance can achieve high recognition rates [24].

Noticing the limited scale invariance in CUREt, researchers from the Royal Institute of Technology (KTH) introduced a dataset called “KTH Textures under varying Illumination, Pose, and Scale” (KTH-TIPS) [82, 174] by imaging ten CUREt materials at three different illuminations, three different poses, and nine different distances, but with significantly fewer settings for lighting and viewing angle than CUREt. KTH-TIPS was created to extend CUREt in two directions: (i) by providing variations in scale (as shown in Fig. 23), and (ii) by imaging different samples of the CUREt materials in different settings. This supports the study of recognizing different samples of the CUREt materials; for instance, does training on CUREt enable good recognition performance on KTH-TIPS? Despite pose variations, KTH-TIPS rotation variations are rather limited.

Experiments with Brodatz or VisTex used different nonoverlapping subregions from the same image for training and testing; experiments with CUREt or KTH-TIPS used different subsets of the images imaged from the identical sample for training and testing. KTH-TIPS2 was one of the first datasets to offer considerable variations within each class. It groups textures not only by instance, but also by the type of material (e.g., wool). It is built on KTH-TIPS and provides a considerable extension by imaging four physical, planar samples of each of eleven materials [174].

The Oulu Texture (Outex) database was collected by the Machine Vision Group at the University of Oulu [162]. It has the largest number of different texture classes (320), with each class
having images photographed under three illuminations and nine rotation angles, but with limited scale variations. Based on Outex, a series of benchmark test suites were derived for evaluations of texture classification or segmentation algorithms [162]. Among them, two benchmark datasets Outex_TC00010 and Outex_TC00012 [163] designated for testing rotation and illumination invariance, appear commonly in papers.

The UIUC (University of Illinois Urbana-Champaign) dataset collected by Lazebnik et al. [110] contains 25 texture classes, with each class having 40 uncalibrated, unregistered images. It has significant variations in scale and viewpoint as well as nonrigid deformations (see Fig. 3 (b)), but has less severe illumination variations than CUReT. The challenges of this database are that there are few sample images per class, but with significant variations within classes. Though UIUC improves over CUReT in terms of large intraclass variations, it is much smaller than CUReT both in the number of classes and the number of images per class. The UMD (University of Maryland) dataset [253] also contains 25 texture classes; similar to UIUC, it has significant viewpoint and scale variations and uncontrolled illumination conditions. As textures are imaged under variable truncation, viewpoint, and illumination, the UIUC and the UMD have stimulated the creation of texture datasets or in a combination of these factors. It has no variations in rotation, viewpoint and scale.

Due to the rapid progress of texture representation approaches, the performance of many methods on the datasets described above are close to saturation, with KTH-TIPS2b being an exception due to its increased complexity. However, most datasets introduced above make the simplifying assumption that textures fill images, and often there is limited intraclass variability, due to a single or limited number of instances, captured under controlled scale, viewpoint and illumination. In recent years, researchers have set their sights on more complex recognition problems where textures appear under poor viewing conditions, low resolution, and in realistic cluttered backgrounds. The Flickr Material Database (FMD) [206, 207] was built to address some of these limitations, by collecting many different object instances from the Internet grouped in 10 different material categories, with examples shown in Fig. 3 (e). The FMD [206] focuses on identifying materials such as plastic, wood, fiber and glass. The limitations of the FMD dataset is that its size is quite small, containing only 10 material classes with 100 images in each class.

The UBO2014 dataset [244] contains 7 material categories, with each having 12 different physical instances. Each material instance was measured by a full bidirectional texture function with

Table 3 Summary of commonly-used texture databases.

| No. | Texture Dataset | References | Total Images | Texture Classes | Image Size | Gray or Color | Imaging Environment | Illumination Changes | Rotation Changes | Viewpoint Changes | Scale Changes | Image Content | Instances or Categories | Year | Download Link |
|-----|----------------|------------|--------------|----------------|------------|--------------|---------------------|--------------------|-------------------|-------------------|---------------|-------------|--------------------------|------|---------------|
| 1   | Brodatz [23]   | [111]      | 111          | 640 × 640     | Gray       | Controlled   | Objects             | Instances          | 1996              | [22]              |              |             |                          |      |               |
| 2   | VisTex         | –          | 167          | 167           | Color      | Wild         | Objects             | Instances          | 1995              | [239]             |              |             |                          |      |               |
| 3   | CUReT [47]     | [5612]     | 92           | 200 × 200     | Color      | Controlled   | Small              | Materials          | 1999              | [43]              |              |             |                          |      |               |
| 4   | Outex [162]    | [8640]     | 320          | 746 × 538     | Color      | Controlled   | √                   | Materials          | 2002              | [171]             |              |             |                          |      |               |
| 5   | KTH-TIPS2 [82, 62] | [810]   | 10           | 200 × 200     | Color      | Controlled   | √                   | Small              | Materials          | 2004              | [104]             |              |             |                          |      |               |
| 6   | UIUC [110]     | [1000]     | 25           | 2500 × 960    | Color      | Wild         | √                   | √                  | Materials          | 2009              | [230]             |              |             |                          |      |               |
| 7   | KTH-TIPS2a [28, 135] | [4608] | 11           | 200 × 200     | Color      | Controlled   | √                   | Small              | Materials          | 2006              | [104]             |              |             |                          |      |               |
| 8   | KTH-TIPS2b [28, 135] | [4752]  | 11           | 200 × 200     | Color      | Controlled   | √                   | Small              | Materials          | 2006              | [104]             |              |             |                          |      |               |
| 9   | UMD [253]      | [1000]     | 25           | 1280 × 960    | Gray       | Wild         | √                   | √                  | √                  | Objects          | 2009              | [230]             |              |                          |      |               |
| 10  | ALOT [25]      | [25000]    | 250          | 1536 × 1024   | Color      | Controlled   | √                   | Materials          | 2009              | [7]               |              |             |                          |      |               |
| 11  | RawFoodT [44]  | [3128]     | 68           | 800 × 800     | Color      | Controlled   | √                   | √                  | √                  | Materials          | 2016              | [193]             |              |                          |      |               |
| 12  | FMD [206, 207] | [1000]     | 10           | 512 × 384     | Color      | Wild         | √                   | √                  | √                  | Materials          | 2009              | [59]              |              |                          |      |               |
| 13  | DreTex [172]   | [40000]    | 20           | 200 × 200     | Color      | Controlled   | √                   | √                  | √                  | √                  | Materials          | 2012              | [53]              |              |                          |      |               |
| 14  | UBO2014 [244]  | [1915284]  | 7            | 400 × 400     | Color      | Synthesis    | √                   | √                  | √                  | Materials          | 2014              | [227]             |              |                          |      |               |
| 15  | OpenSurfaces   | [13]       | 10422        | 22            | Unfixed     | Color        | Wild              | √                   | √                  | √                  | √                  | Materials          | Clutter 2013 | [169]             |              |                          |      |               |
| 16  | DTD [34]       | [5640]     | 47           | Unfixed       | Color      | Wild         | √                   | √                  | √                  | √                  | Attributes        | Categories 2014 | [54]              |              |                          |      |               |
| 17  | MNC [14]       | [2996674]  | 23           | Unfixed       | Color      | Wild         | √                   | √                  | √                  | √                  | Materials          | Clutter 2015 | [156]             |              |                          |      |               |
| 18  | MNC2500 [14]   | [57500]    | 23           | 362 × 362     | Color      | Wild         | √                   | √                  | √                  | √                  | Materials          | Clutter 2015 | [156]             |              |                          |      |               |
| 19  | GTOS [255]     | [34243]    | 40           | 240 × 240     | Color      | Partially    | Controlled        | √                   | √                  | √                  | √                  | Materials          | Instances 2016 | [74]              |              |                          |      |               |
| 20  | LFMD [241]     | [1200]     | 12           | 3787 × 2632   | Color      | Uncontrolled | √                   | √                  | √                  | Materials          | Categories 2016 | [117]             |              |                          |      |               |
| 21  | RDAD [17]      | [1488]     | 57           | 2592 × 1944   | Color      | Uncontrolled | √                   | √                  | Objects            | Instances 2016 | [198]             |              |                          |      |               |
well-sampled dataset such as MINC is key for real-world material recognition and segmentation.

Concurrent to the work by Bell et al. [14], Cimpoi et al. [36] derived a new dataset from OS to conduct a study of material and describable texture attribute recognition in clutter. Since not all segments in OS have a complete set of annotations, Cimpoi et al. [36] selected a subset of segments annotated with material names, annotated the dataset with eleven texture attributes, and removed those material classes containing fewer than 400 segments. Similarly, the Robotics Domain Attributes Database (RDAD) [17] contains 57 categories of everyday indoor object and surface textures labeled with a set of seventeen texture attributes, collected to addresses the target domain of everyday objects and surfaces that a service robot might encounter.

Wang et al. [241] introduced a new light-field dataset of materials, called the Light-Field Material Database (LFMD). Since light-fields can capture multiple viewpoints in a single shot, they implicitly contain reflectance information, which should be helpful in material recognition. The goal of LFMD is to investigate whether 4D light-field information improves the performance of material recognition.

Finally, Xue et al. [255] built a material database named the Ground Terrain in Outdoor Scenes (GTOS) to study the use of spatial and angular reflectance information of outdoor ground terrain for material recognition. It consists of over 30,000 images covering 40 classes of outdoor ground terrain under varying weather and lighting conditions.

### 6.2 Performance

Table 4 presents a performance summary of representative methods applied to popular benchmark texture datasets. It is clear that major improvements have come from more powerful local texture descriptors such as MRELBP [132, 131], ScatNet [24] and CNN-based descriptors [36] and from advanced feature encoding methods like IFV [179]. With the advance in CNN architectures, CNN-based texture representations have quickly demonstrated their strengths in texture classification, especially for recognizing textures with very large appearance variations, such as in KTHTIPS2b, FMD and DTD.

Off-the-shelf CNN based descriptors, in combination with IFV feature encoding, have advantages in nearly all of the benchmark datasets, except for Outex_TC10 and Outex_TC12, where texture descriptors, such as MRELBP [132, 131] and ScatNet, [24] that have rotation and gray scale invariances, give perfect accuracies,
revealing one of the limitations of CNN based descriptors in being sensitive to image degradations. Despite the usual advantages of CNN based methods, it is at a cost of very high computational complexity and memory requirements. We believe that traditional texture descriptors, like the efficient LBP and robust variants such as MRELBP, still have merits in cases when real-time computation is a priority or when robustness to image degradation is needed. [132].

As can be seen from Table 4, currently the highest classification scores on Outex_TC10, Outex_TC12, CUReT, KTHTIPS, UIUC, UMD and ALOT are nearly perfect, in excess of 99.5%, and quite a few texture representation approaches can achieve more than 99.0% accuracy on these datasets. Since the influential work by Cimpoi et al. [34, 35, 36], who reported near perfect classification accuracies with pretrained CNN features for texture classification, subsequent representative CNN based approaches have not reported results on these datasets because performance is saturated and because the datasets are not large enough to allow finetuning to obtain improved results. The FMD, DTD and KTHTIPS2b are undoubtedly more challenging than other texture datasets, for example the UIUC and FMD texture category separation shown in Fig. 27, and these more challenging datasets appear more frequently in recent works. However, since the IFV encoding of VGGVD descriptors [36], the progress on these three datasets has been slow, with incremental improvements in accuracy and efficiency obtained by building more complex or deeper CNN architectures.

As can be observed from Table 4, LBP type methods (LBP [163], MRELBP [131] and BIF [40]) which adopt a predefined codebook have a much more efficient feature extraction step than the remaining methods listed. For those BoW based methods which require codebook learning, since the codebook learning, feature encoding, and pooling process are similar, the distinguishing factors are the computation and feature dimensionality of the local texture descriptor. Among commonly-used local texture descriptors, those approaches first detecting local regions of interest followed by local descriptors, such as SIFT, RIFT and SPIN [110, 263], are among the slowest and have relatively high dimensionality. For the CNN based methods developed in [34, 35, 36], CNN feature extraction is performed on multiple scaled versions of the original texture image, which requires more computational time. In general, CNN pretraining and finetuning is efficient, whereas CNN model training is time consuming. From [132], ScatNet is computationally expensive at the feature extraction stage, though it has medium feature dimensionality. Finally, at the feature classification stage linear SVM is significantly faster than kernel SVM.

7 Discussion and Conclusion

The importance of texture representations lies in the fact that they have extended to many different problems beyond that of textures themselves. As a comprehensive survey on texture representations, this paper has highlighted the recent achievements, provided some structural categories for the methods according to their roles in feature representation, analyzed their merits and demerits, summarized existing popular texture datasets, and discussed performance for the most representative approaches. Almost any practical application is a compromise among conflicting requirements such as classification accuracy, robustness to image degradations, compactness and efficiency, number of training data available, and cost and power consumption of implementations. Although significant progress has been made, the following discussion identifies a number of promising directions for exploratory research.

Large Scale Texture Dataset Collection. The constantly increasing volume of image and video data creates new opportunities and challenges. The complex variability of big image data reveals the inadequacies of conventional handcrafted texture descriptors and brings opportunities for representation learning techniques, such as deep learning, which aim at learning good representations automatically from data. The recent success of deep learning in image classification and object recognition is inseparable from the availability of large-scale annotated image datasets such as ImageNet [201] and MS COCO [121]. However, deep learning based texture analysis has not kept pace with the rapid progress witnessed in other fields, partially due to the unavailability of a large-scale texture database. As a result there is significant motivation for a
good, large-scale texture dataset, which will significantly advance texture analysis.

More Effective and Robust Texture Representations. Despite significant progress in recent years most texture descriptors, irrespective of whether handcrafted or learned, have not been capable of performing at a level sufficient for real world textures. The ultimate goal of the community is to develop texture representations that can accurately and robustly discriminate massive image texture categories in all possible scenes, at a level comparable to the human visual system. In practical applications, factors such as significant changes in illumination, rotation, viewpoint and scale, and image degradations such as occlusions, image blur and random noise call for more discriminative and robust texture representations. Further input from psychological research of visual perception and the biology of the human visual system would be welcome.

Compact and Efficient Texture Representations. There is a tension between the demands of big data and desire for highly compact and efficient feature representations. Thus, on the one hand, many existing texture representations are failing to keep pace with the emerging “big dimensionality” [260], leading to a pressing need for new strategies in dealing with scalability, high computational complexity, and storage. On the other hand, there is a growing need for deploying highly compact and resource-efficient feature representations on platforms like low energy embedded vision sensors and handheld devices. Many of the existing descriptors would similarly fail in these contexts, and the current general trend of deep CNN architectures has been to develop deeper and more complicated networks, advances requiring massive data and power hungry GPUs, not suitable to be deployed on mobile platforms that have limited resources. As a result, there is a growing interest in building compact and efficient CNN-based features [85, 192]. While CNNs generally outperform classical texture descriptors, it remains to be seen which approaches will be most effective in resource-limited contexts, and whether some degree of LBP / CNN hybridization might be considered, such as recent lightweight CNN architectures [124, 251].

Reduced Dependence on Large Amounts of Data. There are many applications where texture representations are very useful and only limited amounts of annotated training data can be available, or where collecting labeled training data is too expensive (such as visual inspection, facial micro-expression recognition, age estimation and medical texture analysis). Possible research could be the development of learnable local descriptors requiring modest training data, as in [55, 136], or to explore effective transfer learning.

Semantic Texture Attributes. Progress in image texture representation and understanding, while substantial, has so far been mostly focused on low-level feature representation. However, in order to address advanced human-centric applications, such as detailed image search and human-robotic interaction, low-level understanding will not be sufficient. Future efforts should be devoted to go beyond texture identification and categorization, to develop semantic and easily describable texture attributes that can be well predicted with low-level texture representations, and to explore even finer-grained and compositional structure analysis of texture patterns.

Effect of Smaller Image Size. Performance evaluation of texture descriptors is usually done with texture datasets consisting of relatively large images. For a large number of applications an ability to analyze small image sizes at high speed is vital, including facial image analysis, interest region description, segmentation, defect detection, and tracking. Many existing texture descriptors would fail in this respect, and it would be important to evaluate the performance of new descriptors [205].

8 Acknowledgments

The authors would like to thank the pioneer researchers in texture analysis and other related fields. The authors would also like to express their sincere appreciation to the associate editor and the reviewers for their comments and suggestions. This work has been supported by the Center for Machine Vision and Signal Analysis at the University of Oulu (Finland) and the National Natural Science Foundation of China under Grant 61872379.

References

1. Aharon M., Elad M., Bruckstein A. (2006) ksvd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans Signal Processing 54(11):4311–4322
2. Ahonen T., Pietikäinen M. (2007) Soft histograms for local binary patterns. In: Proceedings of the Finnish Signal Processing Symposium, vol 5, p. 1
3. Ahonen T., Hadid A., Pietikäinen M. (2006) Face description with local binary patterns: application to face recognition. IEEE TPAMI 28(12):2037–2041
4. Ahonen T., Hadid A., Pietikäinen M. (2006) Face description with local binary patterns: Application to face recognition. IEEE TPAMI 28(12):2037–2041
5. Akl A., Yaacoub C., Donias M., Da Costa J., Germain C. (2018) A survey of exemplar based texture synthesis methods. CVIU
6. Alahi A., Ortiz R., Vanderghynst P. (2012) FREAK: Fast retina keypoint. In: CVPR, pp. 510–517
7. ALOT (2009) http://aloi.science.uva.nl/public_alot/
8. Amadasun M., King R. (1989) Textural features corresponding to textural properties. IEEE Transactions on Systems, Man, and Cybernetics 19(5):1264–1274
9. Andrzejczak V., Whelan P. (2016) Using filter banks in convolutional neural networks for texture classification. Pattern Recognition Letters 84:63–69
10. Arandjelovic R., Gronat P., Torii A., Pajdla T., Sivic J. (2016) NetVLAD: CNN architecture for weakly supervised place recognition. In: CVPR, pp. 5297–5307
11. Baraniuk R., Davenport M., DeVore R., Wakin M. (2008) A simple proof of the restricted isometry property for random matrices. Constructive Approximation 28(3):253–263
12. Bay H., Tuytelaars T., Van Gool L. (2006) SURF: Speeded up robust features. In: ECCV, pp. 404–417
13. Bell S., Upchurch P., Snavely N., Bala K. (2013) Opensurfaces: A richly annotated catalog of surface appearance. ACM Transactions on Graphics 32(4):111
14. Bell S., Upchurch P., Snavely N., Bala K. (2015) Material recognition in the wild with the materials in context database. In: CVPR, pp. 3479–3487
15. Bengio Y., Courville A., Vincent P. (2013) Representation learning: A review and new perspectives. IEEE TPAMI 35(8):1798–1828
16. Bhushan N., Rao A. R., Lohse G. L. (1997) The texture lexicon: Understanding the categorization of visual texture terms and their relationship to texture images. Cognitive Science 21(2):219–246
17. Bornemann R., Esslinger D., Hundsdorfer D., Haegele M., Vincze M. (2016) Texture characterization with semantic attributes: Database and algorithm. In: The 47th International Symposium on Robotics, pp. 1–8
18. Bosch A., Zisserman A., Muñoz X. (2008) Scene classification using a hybrid generative/discriminative approach. IEEE TPAMI 30(4):712–727
19. Bourreau Y., Ponce J., LeCun Y. (2010) A theoretical analysis of feature pooling in visual recognition. In: ICML, pp. 111–118
20. Bovik A., Clark M., Geisler W. (1990) Multichannel texture analysis using localized spatial filters. IEEE TPAMI 12(1):55–73
21. Brahmna S., Jain L., Nanni L., Lumini A. (2014) Local binary patterns: new variants and applications. Springer
22. Brodatz (1966) http://www.ux.uis.no/~tranden/brodatz.htm
23. Brodatz P. (1966) Textures: A Photographic Album for Artists and Designers. Dover Publications, New York, USA
24. Bruna J., Mallat S. (2013) Invariant scattering convolution networks. IEEE TPAMI 35(8):1872–1886
25. Burghouts G., Geusebroek J. (2009) Material specific adaptation of color invariant features. Pattern Recognition Letters 30(3):306 – 313
26. Calonder M., Lepetit V., Ozuysal M., Trzcinski T., Strecha C., Fua P. (2012) BRIEF: Computing a local binary descriptor very fast. IEEE TPAMI 34:1281–1298
27. Candes E. J., Tao T. (2006) Near-optimal signal recovery from random projections: Universal encoding strategies? IEEE Trans Information Theory 52(12):5406–5425
28. Caputo B., Hayman E., Mallikarjuna P. (2005) Class specific material categorisation. In: ICCV, vol 2, pp. 1597–1604
29. Chan T., Jia K., Gao S., Lu J., Zeng Z., Ma Y. (2015) PCANet: A simple deep learning baseline for image classification? IEEE Trans Image Processing 24(12):5017–5032
30. Chatfield K., Lempitsky V., Vedaldi A., Zisserman A. (2011) The devil is in the details: an evaluation of recent feature encoding methods. In: BMVC, vol 2, p. 8
31. Chatfield K., Simonyan K., Vedaldi A., Zisserman A. (2014) Return of the devil in the details: Delving deep into convolutional nets. BMVC
32. Chellappa R., Chatterjee S. (1985) Classification of textures using Gaussian Markov Random fields. IEEE Trans Acoustics, Speech, and Signal Processing 33(4):959–963
33. Chen J., Shan S., He C., Zhao G., Pietikainen M., Chen X., Gao W. (2010) Wild: a robust local image descriptor. IEEE TPAMI 32(9):1705–1720
34. Cimpoi M., Maji S., Kokkinos I., Mohamed S., Vedaldi A. (2014) Describing textures in the wild. In: CVPR, pp. 3606–3613
35. Cimpoi M., Maji S., Vedaldi A. (2015) Deep filter banks for texture recognition and segmentation. In: CVPR, pp. 3828–3836
36. Cimpoi M., Maji S., Kokkinos I., Vedaldi A. (2016) Deep filter banks for texture recognition, description, and segmentation. IJCV 118(1):65–94
37. Cinbis R. G., Verbeek J., Schmid C. (2016) Approximate fisher kernels of non-iid image models for image categorization. IEEE TPAMI 38(6):1084–1098
38. Coates A., Ng A. (2011) The importance of encoding versus training with sparse coding and vector quantization. In: ICML, pp. 921–928
39. Conners R. W., Harlow C. A. (1980) A theoretical comparison of texture algorithms. IEEE TPAMI (3):204–222
40. Crosier M., Griffin L. D. (2010) Using basic image features for texture classification. IJCV 88(3):447–460
41. Cross G., Jain A. (1983) Markov random field texture models. IEEE TPAMI (1):25–39
42. Csurka G., Dance C., Fan L., Willamowski J., Bray C. (2004) Visual categorization with bags of keypoints. In: ECCV Workshop on statistical learning in computer vision
43. CURET (1999) http://www.cs.columbia.edu/CAVE/software/curet/html/about.php
44. Cusano C., Napoliello P., Schettini R. (2016) Evaluating color texture descriptors under large variations of controlled lighting conditions. Journal of the Optical Society of America A 33(1):17–30
45. Dahl A., Larsen R. (2011) Learning dictionaries of discriminative image patches. In: BMVC
46. Dai X., Ng J. Y.-H., Davis L. S. (2017) FASON: First and Second Order information fusion Network for texture recognition. In: CVPR, pp. 7352–7360
47. Dana K., Van Ginneken B., Nayar S., Koenderink J. (1999) Reflectance and texture of real world surfaces. ACM Transactions On Graphics 18(1):1–34
48. Depeursinge A., Al-Kadi O., Mitchell J. (2017) Biomedical Texture Analysis. Academic Press
49. Ding C., Choi J., Tao D., Davis L. S. (2016) Multi-directional multi-level dual-cross patterns for robust face recognition. IEEE TPAMI 38(3):518–531
50. Dixit M., Chen S., Gao D., Rasiwasia N., Vasconcelos N. (2015) Scene classification with semantic fisher vectors. In: CVPR, pp. 2974–2983
51. Dixit M. D., Vasconcelos N. (2016) Object based scene representations using fisher scores of local subspace projections. In: NIPS, pp. 2811–2819
52. Donoho D. L. (2006) Compressed sensing. IEEE Trans Information Theory 52(4):1289–1306
53. Drexel (2012) https://www.cs.drexel.edu/~kon/textrue/
54. DTD (2014) http://www.robots.ox.ac.uk/~vgg/data/dtd/
55. Duan Y., Lu J., Feng J., Zhou J. (2017) Context aware local binary feature learning for face recognition. IEEE TPAMI
56. Efros A. A., Leung T. K. (1999) Texture synthesis by nonparametric sampling. In: ICCV, vol 2, pp. 1033–1038
57. Everingham M., Eslami S., Gool L. V., Williams C., Winn J., Zisserman A. (2015) The pascal visual object classes challenge: A retrospective. IJCV 111(1):98–136
58. Farhadi A., Endres I., Hoiem D., Forsyth D. (2009) Describing objects by their attributes. In: CVPR, pp. 1778–1785
59. FMD (2009) http://people.csail.mit.edu/ cellu1/cvpr2010/FMD/
60. Forsyth D., Ponce J. (2012) Computer Vision: A Modern Approach, 2nd Edition
61. Freeman W., Adelson E. (1991) The design and use of steerable filters. IEEE TPAMI 13(9):891–906
62. Fritz M., Hayman E., Caputo B., Eklundh J. (2004) The KTH-TIPS database. http://www.nada.kth.se/cvap/databases/kth-tips/kth-tips.pdf
63. Gao Y., Beijbom O., Zhang N., Darrell T. (2016) Compact bilinear pooling. In: CVPR, pp. 317–326
64. Gärden J., Lindeberg T. (1996) Direct computation of shape cues using scale-adapted spatial derivative operators. IJCV 17(2):163–191
65. Gatys L., Ecker A., Bethge M. (2015) Texture synthesis using convolutional neural networks. In: NIPS, pp. 262–270
66. Gatys L., Ecker A., Bethge M. (2016) Image style transfer using convolutional neural networks. In: CVPR, pp. 2414–2423
67. Georgescu B., Shimshoni I., Meer P. (2003) Mean shift based clustering in high dimensions: A texture classification example. In: ICCV, vol 3, p. 456
68. Girshick R., Donahue J., Darrell T., Malik J. (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR, pp. 580–587
69. Giryes R., Sapiro G., Bronstein A. M. (2016) Deep neural networks with random gaussian weights: a universal classification strategy? IEEE Trans Signal Processing 64(13):3444–3457
70. Gong Y., Wang L., Guo R., Lazebnik S. (2014) Multi scale orderless pooling of deep convolutional activation features. In: ECCV, pp. 392–407
71. Grauman K., Darrell T. (2005) The pyramid match kernel: Discriminative classification with sets of image features. In: ICCV, vol 2, pp. 1458–1465
72. Griffin L., Lillholm M., Crosier M., van Sande J. (2009) Basic image features (bifs) arising from approximate symmetry type. Scale Space and Variational Methods in Computer Vision pp. 343–355
73. Griffin L. D., Lillholm M. (2010) Symmetry sensitivities of derivative-of-gaussian filters. IEEE TPAMI 32(6):1072–1083
74. Gu J., Wang Z., Kuen J., Ma L., et al. (2017) Recent advances in convolutional neural networks. Pattern Recognition
75. Guo Z., Zhang L., Zhang D. (2010) A completed modeling of local binary pattern operator for texture classification. IEEE Trans Image Processing 9(16):1557–1563
76. Guo Z., Wang X., Zhou J., You J. (2016) Robust texture image representation by scale selective local binary patterns. IEEE Trans Image Processing 25(2):687–699
78. Han J., Ma K. (2007) Rotation invariant and scale invariant gabor features for texture image retrieval. Image and Vision Computing 25(9):1474–1481
79. Haralick R. (1979) Statistical and structural approaches to texture. Proceedings of the IEEE 67(5):786–804
80. Haralick R., Shanmugam K., Dinstein I. (1973) Textural features for image classification. IEEE Trans on Systems, Man, and Cybernetics (6):610–621
81. Hariharan B., Arbeláez P., Girshick R., Malik J. (2015) Hypercolumns for object segmentation and fine-grained localization. In: CVPR, pp. 447–456
82. Hayman E., Caputo B., Fritz M., Eklundh J. (2004) On the significance of real world conditions for material classification. ECCV pp. 253–266
83. He C., Li S., Liao Z., Liao M. (2013) Texture classification of PolSAR data based on sparse coding of wavelet polarization textures. IEEE Trans Geoscience and Remote Sensing 51(8):4576–4590
84. He K., Zhang X., Ren S., Sun J. (2016) Deep residual learning for image recognition. In: CVPR, pp. 770–778
85. Howard A., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: CVPR
86. Huang D., Shan C., Ardabilian M., Wang Y., Chen L. (2011) Local binary patterns and its application to facial image analysis: a survey. IEEE Transactions on Systems, Man, and Cybernetics Part C 41(6):765–781
87. Huang G., Liu Z., Weinberger K. Q., van der Maaten L. (2017) Densely connected convolutional networks. In: CVPR
88. Huang Y., Wu Z., Wang L., Tan T. (2014) Feature coding in image classification: A comprehensive study. IEEE TPAMI 36(3):493–506
89. Jain A., Farrokhnia F. (1991) Unsupervised texture segmentation using Gabor filters. Pattern Recognition 24(12):1167–1186
90. Jain A., Duin R., Mao J. (2000) Statistical pattern recognition: A review. IEEE TPAMI 22(1):4–37
91. Jegou H., Perronnin F., Douze M., Sánchez J., Perez P., Schmid C. (2012) Aggregating local image descriptors into compact codes. IEEE TPAMI 34(9):1704–1716
92. Julesz B. (1962) Visual pattern discrimination. IRE transactions on Information Theory 8(2):84–92
93. Julesz B. (1981) Textons, the elements of texture perception, and their interactions. Nature 290(5802):91–97
94. Julesz B., Bergen J. (1983) Human factors and behavioral science: Textons, the fundamental elements in preattentive vision and perception of textures. The Bell System Technical Journal 62(6):1619–1645
95. Kadir T., Brady J. (2002) Scale, saliency and scene description. PhD thesis, Oxford University
96. Kandaswamy U., Adjeroh D., Lee M. (2005) Efficient texture analysis of SAR imagery. IEEE Trans Geoscience and Remote Sensing 43(9):2075–2083
97. Kandaswamy U., Schuckers S., Adjeroh D. (2011) Comparison of texture analysis schemes under nonideal conditions. IEEE Trans Image Processing 20(8):2260–2275
98. Keller J., Chen S., Crowonover R. (1989) Texture description and segmentation through fractal geometry. Computer Vision, Graphics, and Image Processing 45(2):150–166
99. Kim K., Jung K., Park S., Kim H. (2002) Support vector machines for texture classification. IEEE TPAMI 24(11):1542–1550
100. Kong S., Fowlkes C. (2017) Low rank bilinear pooling for fine-grained visual recognition. In: CVPR, pp. 1265–1278
101. Kong S., Schmid C., Ponce J. (2005) A sparse texture representation using local affine regions. IEEE TPAMI 27(8):1265–1278
102. Krishna R., Zhu Y., Groth O., Johnson J., Hata K., Kravitz J., Chen S., Kalantidis Y., Li L., Shamma D., Bernstein M., Fei-Fei L. (2017) Visual genome: Connecting language and vision using crowdsourced dense image annotations. IJCV 123(1):32–73
103. Krizhevsky A. (2004) ImageNet classification with deep convolutional neural networks. In: NIPS, pp. 1097–1105
104. KTHTIPS (2004) http://www.nada.kth.se/cvap/databases/kth-tips/download.html
105. Kumar N., Berg A., Belhumeur P. N., Nayar S. (2011) Describable visual attributes for face verification and image search. IEEE TPAMI 33(10):1962–1977
106. Kwitt R., Vasconcelos N., Rasiwasia N. (2012) Scene recognition on the semantic manifold. In: ECCV, Springer, pp. 359–372
107. Lategahn H., Gross S., Stehle T., Aach T. (2010) Texture classification by modeling joint distributions of local patterns with gaussian mixtures. IEEE Trans Image Processing 19(6):1548–1557
108. Laws K. (1980) Rapid texture identification. In: Proc. SPIE Conf. Image Processing for Missile Guidance, vol 238, pp. 376–381
109. Leutenegger S., Chili M., Siegwart R. (2011) BRISK: Binary robust invariant scalable keypoints. In: ICCV, pp. 2486–2493
110. Leutenegger S., Chili M., Siegwart R. (2011) Emotion recognition in the wild via convolutional neural networks and mapped binary patterns. In: ACM ICMI, pp. 303–310
111. Levi G., Hassner T. (2015) Emotion recognition in the wild via convolutional neural networks. In: CVPR, pp. 2169–2178
112. Li L. (2014) Object bank: An object level image representation for high level visual recognition. IJCV 107(1):20–39
113. Li L., Ma K. (2007) Rotation invariant scalable keypoints. In: CVPR, vol 2, pp. II–II
114. Li L., Fieguth P., Wu Z., Tan T. (2014) Feature coding in image classification. IEEE TPAMI 36(3):493–506
115. Lin T., RoyChowdhury A., Maji S. (2015) Bilinear cnn models for fine-grained visual recognition. In: CVPR, pp. 1449–1457
116. Lin T., RoyChowdhury A., Maji S. (2017) Bilinear convolutional neural networks for fine-grained visual recognition. IEEE TPAMI
117. Lin X., Zhao C., Pan W. (2017) Towards accurate binary convolutional neural network. In: NIPS, pp. 344–352
118. Liu L., Fieguth P. (2012) Texture classification from random features. IEEE TPAMI 34(3):574–586
119. Liu L., Fieguth P., Wang L., Zha H. (2011) Sorted random projections for robust texture classification. In: ICCV, IEEE, pp. 391–398
120. Liu L., Wang L., Liu X. (2011) In defense of soft assignment coding. In: ICCV, pp. 2486–2493
121. Liu L., Fieguth P., Wang L., Zha H. (2011) Sorted random projections for robust rotation invariant texture classification. Pattern Recognition 45(6):2405–2418
122. Liu L., Fieguth P., Hu D., Wei Y., Wang J. (2015) Fusing sorted random projections for robust texture and material classification. IEEE TCSVT 25(3):482–496
123. Liu L., Fieguth P., Wang X., Pietikäinen M., Hu D. (2016) Evaluation of LBP and deep texture descriptors with a new robustness benchmark. In: ECCV
124. Liu L., Liao S., Fieguth P., Guo Y., Wang X., Pietikäinen M. (2016) Median robust extended local binary pattern for texture classification. IEEE Trans Image Processing 25(3):1368–1381
125. Liu L., Fieguth P., Guo Y., Wang X., Pietikäinen M. (2017) Local binary features for texture classification: Taxonomy and experimental study. Pattern Recognition 62:135–160
126. Liu L., Liao S., Fieguth P., Guo Y., Wang X., Pietikäinen M. (2017) Local binary features for texture classification: Taxonomy and experimental study. Pattern Recognition 62:135–160
127. Liu L., Ouyang W., Wang X., Fieguth P., Chen J., Liu X., Pietikäinen M. (2018) Deep learning for generic object detection: A survey. arXiv:180902165
128. Liu Y., Tsin Y., Lin W. (2005) The promise and perils of near regular texture. IJCV 62(1):145–159
129. Lowe D. (2004) Distinctive image features from scale-invariant key-points. IJCV 60(2):91–110
195. Sifre L., Mallat S. (2012) Combined scattering for rotation invariant texture recognition. Vision Research 44(19):2301–2311

196. Sharan L., Rosenholtz R., Adelson E. (2013) Recognizing materials using perceptually inspired features. IJCV 103(3):348–371

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

198. Schmid C. (2001) Constructing models for content based image retrieval. In: CVPR, vol 2, pp. 39–45

199. Rublee E., Rabaud V., Konolige K., Bradski G. (2011) ORB: An efficient alternative to SIFT or SURF. In: ICCV, pp. 2564–2571

200. Rubner Y., Tomasi C., Guibas L. (2000) The Earth Mover’s Distance as a metric for image retrieval. IJCV 40(2):99–121

201. Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., et al. (2015) ImageNet large scale visual recognition challenge. IJCV 115(3):211–252

202. Ryu J., Hong S., Yang H. (2015) Sorted consecutive local binary pattern for texture classification. IEEE Trans Image Processing 24(7):2254–2265

203. Sanchez J., Perronnin F., Mensink T., Verbeek J. (2013) Image classification with the fisher vector: Theory and practice. IJCV 105(3):222–245

204. Schwartz G., Nishino K. (2015) Automatically discovering local visual material attributes. In: IJCV, pp. 3565–3573

205. Schwartz G., Nishino K. (2015) Equivalence of julesz ensembles and multifractal analysis in multiorientation wavelet pyramid. In: CVPR, pp. 1–11

206. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Locally transferred fisher vectors for texture classification. In: CVPR, pp. 4912–4920

207. Sivic J., Zisserman A. (2005) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

208. Sivic J., Zisserman A. (2003) Video google: A text retrieval approach to video analysis. In: Proc. European Symp. Artificial Neural Networks, pp. 125–130

209. Sivic J., Zisserman A. (2005) A statistical approach to texture classification from single images. IJCV 62(1-2):61–81

210. Sokolin A., Reinhard E., Efros A. A. (2015) A 4d light field dataset and cnn architectures for material recognition. In: CVPR, pp. 121–138

211. Solomon J., Durand F., Sivic J. (2012) Works: Maximizing quality and diversity in feed forward stylization and texture synthesis. In: International Conference on Computer Vision and Pattern Recognition

212. Solomon J., Durand F., Sivic J. (2012) Texture synthesis. In: International Conference on Computer Vision and Pattern Recognition.

213. Solomon J., Durand F., Sivic J. (2012) Imagery/VisionTexture/texture.html

214. Solomatine D. (2011) A new texture descriptor using multi Fractal analysis. Pattern Recognition

215. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

216. Sivic J., Zisserman A. (2005) A statistical approach to texture classification from single images. IJCV 62(1-2):61–81

217. Solomatine D. (2011) A new texture descriptor using multi Fractal analysis. Pattern Recognition

218. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

219. Solomatine D. (2011) A new texture descriptor using multi Fractal analysis. Pattern Recognition

220. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

221. Solomatine D. (2011) A new texture descriptor using multi Fractal analysis. Pattern Recognition

222. Sivic J., Zisserman A. (2005) A statistical approach to texture classification from single images. IJCV 62(1-2):61–81

223. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

224. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

225. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

226. Sivic J., Zisserman A. (2005) A statistical approach to texture classification from single images. IJCV 62(1-2):61–81

227. Solomatine D. (2011) A new texture descriptor using multi Fractal analysis. Pattern Recognition

228. Song Y., Zhang F., Li Q., Huang H., O’Donnell L., Cai W. (2017) Local invariant fractal features for statistical texture classification. In: ICCV, pp. 1–8

229. Solomatine D. (2011) A new texture descriptor using multi Fractal analysis. Pattern Recognition

230. UMD (2009) http://www.cfar.umd.edu/~fer/
256. Yang J., Yu K., Gong Y., Huang T. (2009) Linear spatial pyramid matching using sparse coding for image classification. In: CVPR, pp. 1794–1801
257. Yang L., Jin R., Sukthankar R., Jurie F. (2008) Unifying discriminative visual codebook generation with classifier training for object category recognition. In: CVPR, pp. 1–8
258. Ylioinas J., Hong X., Pietikäinen M. (2013) Constructing local binary pattern statistics by soft voting. In: Scandinavian Conference on Image Analysis, pp. 119–130
259. Zhai H., Liu C., Dong H., Ji Y., Guo Y., Gong S. (2015) Face verification across aging based on deep convolutional networks and local binary patterns. In: International Conference on Intelligent Science and Big Data Engineering, pp. 341–350
260. Zhai Y., Ong Y.-S., Tsang I. (2014) The emerging “big dimensionality”. IEEE Computational Intelligence Magazine 9(3):14–26
261. Zhang H., Jia X., Dana K. (2017) Deep TEN: Texture encoding network. In: CVPR
262. Zhang J., Tan T. (2002) Brief review of invariant texture analysis methods. Pattern recognition 35(3):735–747
263. Zhang J., Marszalek M., Lazebnik S., Schmid C. (2007) Local features and kernels for classification of texture and object categories: a comprehensive study. IJCV 73(2):213–238
264. Zhang W., Shan S., Gao W., Chen X., Zhang H. (2005) Local gabor binary pattern histogram sequence (LGBPHS): A novel nonstatistical model for face representation and recognition. In: ICCV, vol 1, pp. 786–791
265. Zhao G., Pietikäinen M. (2007) Dynamic texture recognition using local binary patterns with an application to facial expressions. IEEE TPAMI 29(6):915–928
266. Zheng L., Yang Y., Tian Q. (2017) SIFT meets CNN: A decade survey of instance retrieval. IEEE TPAMI
267. Zhou B., Lapiedra A., Xiao J., Torralba A., Oliva A. (2014) Learning deep features for scene recognition using places database. In: NIPS, pp. 487–495
268. Zhou B., Lapiedra A., Khosla A., Oliva A., Torralba A. (2017) Places: A 10 million image database for scene recognition. IEEE TPAMI
269. Zhu S. (2003) Statistical modeling and conceptualization of visual patterns. IEEE TPAMI 25(6):691–712
270. Zhu S., Wu Y., Mumford D. (1998) Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. IJCV 27(2):107–126
271. Zhu S., Liu X., Wu Y. (2000) Exploring texture ensembles by efficient markov chain monte carlo-toward a “trichromacy” theory of texture. IEEE TPAMI 22(6):554–569
272. Zhu S., Guo C., Wang Y., Xu Z. (2005) What are textons? IJCV 62(1):121–143
Table 4: Performance (%) summarization of some representative methods on popular benchmark texture datasets. All methods used the same splitting strategy for training and testing on each dataset. Specifically, for KTHTIPS2, one image per class is used for training and the remaining three for testing. For Brodatz, please see [110]; for Outex, TC10 and Outex, TC12 please see [130]. For DTD, 80 images per class are randomly selected for training and the remaining 40 for testing. For all other datasets, half of the samples per class are chosen for training and the remaining half for testing. Results are averaged over a number of random partitions of training and testing data. All listed results are quoted from the original papers, except those marked with (*) from [263], and those marked (―) from [130]. For interested readers, more results on LB9 variants can be found in the recent survey [130, 132]. Those dimensions with (†) denote feature dimension before the SoftMax layer. For Brodatz, KTHTIPS2, FMD and DTD, the highest classification score is highlighted; for all other datasets classification scores higher than 99% are highlighted.

| Method Info | Texture Representation and Classification | Results Reported on Popular Benchmark Texture Datasets |
|-------------|----------------------------------------|------------------------------------------------------|
| **Method** | **Published in** | **In Described** | **Texture Representation** | **Texture Classification** | **Texture Dimension** |
| **LBP [163]** | TAPM 2002 | Section 3.2.2 | LBP○○○○ ○○○○ | predefined BoW | Chi Square, NCC | 210 |
| **MR8 [256]** | IJCV 2005 | Section 3.2.2 | MR8 filters | means | BoW | Chi Square, NCC | 2440 |
| **MR8 [82]** | ICCV 2004 | Section 3.2.2 | MR8 filters | means | BoW | Chi Square, SVM | 2440 |
| **Lazebnik et al. [130]** | TPAMI 2005 | Section 3.2.1 | GIST, BoQs, SPIN, RPF | means | Signatures | EMD, NCC | 40 Clusters |
| **Zhang et al. [263]** | IJCV 2007 | Section 3.2.1 | GIST, BoQs, SPIN, RPF, SIFT | means | Signatures | EMD, SVM | 40 Clusters |
| **MFS [253]** | IJCV 2009 | Section 3.2.2 | GIST, Energy Laplacian | MFS Pooling | l1, NCC | 78 |
| **OFF [252]** | CVPR 2009 | Section 3.2.2 | Multiscale Orientation Hist. | Multiscale Decomposition of MFS Vectors | RBF SVM | 1160 |
| **WMFS [254]** | CVPR 2010 | Section 3.2.2 | Multiresolution Wavelet Pyramid | MFS Pooling | RBF SVM | 103 |
| **Patch [237]** | TPAMI 2009 | Section 3.2.2 | Patch Vectors | means | BoW | Chi Square, NCC |
| **BIF [140]** | IJCV 2010 | Section 3.2.2 | BIF Features | means | predefined BoW | Chi Square, NCC | 1292 |
| **RP [125]** | TPAMI 2012 | Section 3.2.2 | Random Features | means | BoW | Chi Square, NCC |
| **SRP [126]** | IJCV 2011 | Section 3.2.2 | SRP Features | means | BoW | Chi Square, SVM |
| **Timofte et al. [223]** | BMVC 2012 | Section 3.2.2 | BIF Features | predefined Multiscale BoW | Reasoning Collaborative | 1780 |
| **Sharan et al. [67]** | IJCV 2013 | Section 3.2.2 | Eight Features (Including SIFT) | means | BoW | Hist Intersection SVM | 1550 |
| **MRELBP [131]** | TP 2016 | Section 3.2.2 | MRELBP○○○○○○ | predefined BoW | Chi Square, SVM | 800 |
| **SIFT [135, 34]** | IJCV 2016 | Section 3.2.2 | Dense SIFT | GMM | IFV | Linear SVM | 65536 |
| **ScatNet [E4, 233]** | TPAMI 2013 | Section 4.3 | Gaussian Wavelet ScatNet | Gaussian Smoothing Average Pooling | PCA Classifier | 596 |
| **PCAnet [29]** | TP 2015 | Section 4.3 | PCAnet Stage 2 | Multilayer LBP Pooling | Linear SVM | 32768 |
| **AlexNet [36]** | IJCV 2016 | Section 4.1 | Pretrained CONV features (Multiscale Inputs) | GMM | IFV | Linear SVM | 65536 |
| **VGGM [36]** | IJCV 2016 | Section 4.1 | Pretrained CONV features | GMM | IFV | Linear SVM | 65536 |
| **VGVG [36]** | IJCV 2016 | Section 4.1 | Pretrained FC features | GMM | IFV | Linear SVM | 65536 |
| **BCNN [120]** | CVPR 2016 | Section 4.1 | VGGCONV | CONV features | Bilinear Pooling | Linear SVM | 262144 |
| **LVFCNN [218]** | CVPR 2017 | Section 4.1 | CONV features | GMM | IFV | Linear SVM | 65536 |
| **ResNet [201]** | CVPR 2017 | Section 4.1 | ResNet50 | GMM | IFV | Linear SVM | 65536 |
| **TCNN [9]** | PRL 2017 | Section 4.2 | AKNet CONV layers | Global Average Pooling, FC, SoftMax | 4096(1) |
| **BCNN [63]** | CVPR 2016 | Section 4.2 | VGGVD CONV layers | Compact Bilinear Pooling (CBP), SoftMax | 8192(1) |
| **FASON [46]** | CVPR 2017 | Section 4.2 | VGGVD CONV layers | CBP and Global Avg. Pooling, SoftMax | 9216(1) |
| **DeepTEN [261]** | CVPR 2017 | Section 4.2 | ResNet50 | Texture Encoding Layer, SoftMax | 4096(1) |