[[1]](#footnote-0)

Texture Classification Using Pair-wise Difference Pooling Based Bilinear Convolutional Neural Networks

Xinghui Dong, Huiyu Zhou, and Junyu Dong

***Abstract*—Texture is normally represented by aggregating local features based on the assumption of spatial homogeneity. Effective texture features are always the research focus even though both hand-crafted and deep learning approaches have been extensively investigated. Motivated by the success of Bilinear Convolutional Neural Networks (BCNNs) in fine-grained image recognition, we propose to incorporate the BCNN with the Pair-wise Difference Pooling (i.e. BCNN-PDP) for texture classification. The BCNN-PDP is built on top of a set of feature maps extracted at a convolutional layer of the pre-trained CNN. Compared with the outer product used by the original BCNN feature set, the pair-wise difference not only captures the pair-wise relationship between two sets of features but also encodes the difference between each pair of features. Considering the importance of the gradient data to the representation of image structures, we further generalise the BCNN-PDP feature set to two sets of feature maps computed from the original image and its gradient magnitude map respectively, i.e. the Fused BCNN-PDP (F-BCNN-PDP) feature set. In addition, the BCNN-PDP can be applied to two different CNNs and is referred to as the Asymmetric BCNN-PDP (A-BCNN-PDP). The three PDP-based BCNN feature sets can also be extracted at multiple scales. Since the dimensionality of the BCNN feature vectors is very high, we propose a new yet simple Block-wise PCA (BPCA) method in order to derive more compact feature vectors. The proposed methods are tested on seven different datasets along with 21 baseline feature sets. The results show that the proposed feature sets are superior, or at least comparable, to their counterparts across different datasets.**

***Index Terms*—Texture classification, texture features, texture, CNNs, BCNNs**

# INTRODUCTION

T

EXTURE is normally treated as a spatial phenomenon even if there is not a formal definition in the literature. The appearance of texture can be associated with the intensity (or colour) variations in the spatial domain. In this context, texture can be divided into regular, near-regular and stochastic textures. No matter what type of textures are considered, spatial homogeneity is always manifested. In other words, texture presents certain repetitive characteristic. Therefore, texture features are normally designed based on the orderless aggregation of local texture characteristic descriptions [19].

In the past forty years, many texture features [10], [23], [29], [34], [37], [39-40], [42], [47-48] have been developed in order to represent the patterns shown in textures. Dong *et al*. [19], [25], [23] surveyed 51 different sets of texture features using four categories, including signal processing based [39], statistical [29], [47], structural [34], [42], [48] and model-based [10], [40]. These features were generally hand-crafted based on some hypothesis on texture characteristics. Since different texture datasets contain different types of textures, the performance of hand-crafted features usually varies across datasets.

In recent years, Convolutional Neural Networks (CNNs) [9], [32], [44] have been learnt from a large dataset containing a huge number of images. Benefiting from a large training set and the powerful learning ability of CNNs, these networks have achieved great progress in many computer vision applications. Although small image datasets in some particular domains, e.g. [21], [53], cannot be used to train a CNN from scratch, it has been demonstrated that features extracted from a pre-trained CNN can be generalised to different domain-specific datasets [43]. In particular, the features extracted at a convolutional layer of a pre-trained CNN are more domain-independent than the fully-connected features extracted from the same CNN [11].

As a common practice, the deep convolutional features can be used to learn visual words. Different word encoders, such as Bag-of-Words (BoW) [45], Fisher Vector (FV) [31], Graph-of-Words (GoW) [21], Spatial Layout of Words (SLoW) [22], Spatial Pyramid Pooling (SPM) [33] and Vector of Locally Aggregated Descriptors (VLAD) [30], have been used to model the distributions of these deep words. Compared with the words learnt from hand-crafted features, e.g. Histogram of Oriented Gradients (HoG) [15], image patches [48] and Scale-Invariant Feature Transform (SIFT) [38], the deep words produced better results with large margins [11].

On the other hand, Bilinear Convolutional Neural Networks (BCNNs) have been proposed by adding an outer product (i.e. pair-wise product) layer and an average pooling layer [36]. In contrast to the abovementioned visual word encoders, BCNNs do not require a dictionary learning stage while they have produced better results in fine-grained image recognition tasks [36]. This is attributed to the fact that BCNNs take into account the pair-wise (second-order) relationship [7] between a set of powerful deep convolutional features.

In essence, the outer product models the pair-wise relationship between two sets of features. However, it does not consider the difference between the two sets of features. Many features have been developed based on the difference statistics, such as grey level difference histograms [47], [50] and local derivative patterns [55]. We are inspired to replace the outer product layer by a pair-wise difference layer. Compared with the outer product, the pair-wise difference not only encodes the pair-wise relationship between two sets of features but also measures the difference between each pair of features. Therefore, the BCNN feature set based on the pair-wise difference is likely to own more discriminant ability than the outer product based BCNN method [36]. The proposed feature set produces a feature vector by applying the pair-wise difference pooling to a set of feature maps computed at a convolutional layer of a pre-trained CNN for an image. Particularly, we introduce an efficient algorithm to fulfil the computation. The feature set is referred to as the “Pair-wise Difference Pooling (PDP) Based BCNN” or “BCNN-PDP”.

Furthermore, the BCNN-PDP is popularised to two sets of feature maps calculated from the original image and the corresponding gradient magnitude map, respectively, due to the fact that the gradient data retains the image structure. Although contours can also be used to represent image structures, they may not be accurately extracted from some images. In contrast, the gradient map can be computed straightforwardly. In essence, the two sets of feature maps are fused into a single feature vector. Thus, this PDP-based BCNN is named Fused BCNN-PDP (F-BCNN-PDP). As Lin *et al.* [36] did, we also apply the PDP-based BCNN to two sets of feature maps calculated from the same image using two different CNNs respectively. This dual-CNN feature set is referred to as the Asymmetric BCNN-PDP (A-BCNN-PDP).

The previous work has shown that multiple resolutions or scales are able to boost the performance of features [11], [20], [25]. Therefore, we propose a multi-scale scheme for the three proposed PDP-based BCNN feature sets based on an Average Scale-wise Pooling. The pooling is performed as the average of the pair-wise difference poolings conducted at each scale.

A significant characteristic of the BCNN [36] methods is that the dimensionality of their feature vectors is high. To reduce the dimensionality of these feature vectors, we also propose a simple Block-wise PCA (BPCA) method. This method first applies an individual PCA operation to a block of the square matrix of the BCNN features, and then collapses the output feature vectors into a single feature vector. In contrast to the PCANet [8] that uses PCA to generate filters, we employ this method for obtaining compact features.

We evaluate the proposed BCNN-PDP, F-BCNN-PDP and A-BCNN-PDP feature sets extensively using sevendifferent image datasets together with 21 baseline feature sets, including hand-crafted and CNN-based feature sets.

The contributions of this study can be summarised as follows.

(1) We propose a new bilinear CNN method based on the pair-wise difference (BCNN-PDP) and its asymmetric version (A-BCNN-PDP).

(2) We introduce a fused BCNN-PDP feature set (F-BCNN-PDP) by jointly using the original image and its gradient magnitude map.

(3) We adapt a multi-scale scheme for the three proposed PDP-based BCNN feature sets. This scheme allows the proposed feature sets to exploit multiple scales and boosts their performance.

(4) We develop a Block-wise PCA (BPCA) method, which effectively reduces the dimensionality of the BCNN feature vectors.

(5) We evaluate the proposed methods on six different texture datasets and an aerial scene dataset along with 21 baselines. The results provide the community with benchmarks for further research.

In the rest of this paper, we first review the related work in Section II. Then, we introduce the proposed methods in Section III. In Section IV, the experimental setup is described. Our main experimental results are reported in Sections V while the BCNN-PDP is further examined in Section VI in detail. In Section VII, we generalise the proposed methods into object recognition. Finally, our conclusions and future work are discussed in Section VIII.

# Related Work

## Texture Features

Dong *et al*. [19], [23], [25] divided texture features into four categories: signal processing based [39], statistical [29], structural [42] and model-based [40]. Using these categories, they surveyed 51 different sets of texture features. Normally, signal processing based features can be derived using the energy (or variance) of the filter responses generated by applying a single filter or a set of filters to an image, e.g. Gabor Wavelet [39]. Many statistical features were designed by encoding the spatial distribution of grey levels, such as gray level co-occurrence matrices (GLCM) [29], absolute gray level difference histograms (GLADH) [50], signed gray level difference histograms (GLSDH) [47], and so on. In the opinion of the structural analysis of textures [49], textures comprise primitives that are placed according to some spatial placement rules. One of the most popular structural texture features is the texton-based feature [34], [48]. In particular, the well-known “Bag-of-Words” (BoW) or “Bag-of-Visual-Words” [45] features belong to the texton-based type. Similarly, local binary patterns (LBP) [42] can also be considered as structural features. Furthermore, several texture models were introduced in order to represent textures, for example, fractal models [10] and simultaneous autoregressive models [40].

Although the aforementioned hand-crafted texture features may produce promising results on some texture datasets, their performance usually varies significantly across different datasets. One possible reason is that these features were designed based on different assumptions. In contrast, features extracted using the pre-trained CNNs, i.e. those networks trained from a huge number of images, can be generalised to different small domain-specific datasets, e.g. [21], [53]. Compared with features derived from the CNNs which were end-to-end trained [51], [56], they are more suitable for the scenario where only small datasets are available. Therefore, we are motivated to employ pre-trained CNNs in this study.

## Texture Classification

Image classification aims at classifying an image into a certain class according to its visual content. Similarly, texture classification assigns one of a set of texture class labels to an unknown texture image [48] or image patch [42]. Texture classification algorithms normally contain two modules: feature extractor and classifier. A feature extractor is first used to compute features (see the above subsection) from an image or image patch in order to describe its visual content. Then, a classifier is used to divide different images or image patches into a set of known classes. Popular classifiers include Decision Trees [14], Naive Bayes [17], Random Forests [4] and Support Vector Machines (SVM) [26].

Most recently, CNNs [9], [32], [44] have dominated the field of image classification. Compared with the datasets used for image classification, texture datasets [1-2], [5], [28] normally contain much fewer images. Insufficient training images prevent one from training such a CNN from scratch. In addition, it was demonstrated that new CNNs should be designed in order to learn spatial correlation based features for describing textures [3]. However, it has been shown that the pre-trained CNNs obtained using a huge number of training images can be used as a generic feature extractor [43]. The features extracted using these CNNs along with a traditional classifier normally produce state-of-the-art results [11], [21-22].

## Application of Pre-trained CNNs

Donahue *et al*. [18] investigated whether or not features computed from the CNN, which had been trained using a large dataset for image recognition, can be generalised to other computer vision applications. It has been shown that these features own the good generalisation ability and produced state-of-the-art results in different tasks together with a linear classifier. Also, Razaviant *et al*. [43] found that the features extracted from the fully-connected layer of a pre-trained CNN achieved state-of-the-art performance in different tasks.

When only small domain-specific datasets are available, the features extracted at one or multiple convolutional layers of a pre-trained CNN can also be used as local features to learn visual words [11], [21-22], [41], [52]. Different word encoders, such as BoW [45], FV [31], GoW [21], SLoW [22], SPM [33] and VLAD [30], have been used to represent the distribution of these words. One of most significant findings of these studies is that the deep visual word features outperformed their counterparts computed using traditional local features and the same word encoder [11], [21-22]. In addition, these features normally produce better results than the fully-connected features extracted using the same pre-trained CNN when they are transferred to small datasets. However, the visual word features require a word dictionary learning process. In contrast, the Bilinear Convolutional Neural Network (BCNN) features [36], [55] which can be computed using a pre-trained CNN based on the outer product or the Hadamard product pooling do not need dictionary learning, and performed better than, or comparably to, the visual word features in fine-grained image recognition. Inspired by the BCNN methods, we propose to use the pair-wise difference pooling for texture classification on small datasets, particularly.

# The Pair-wise Difference Pooling Based Bilinear Convolutional Neural Networks (BCNNs-PDP)

The Pair-wise Difference Pooling (PDP) is applied to the bilinear CNN (BCNN) [36] by replacing the outer product pooling. In contrast to the outer product, the pair-wise difference not only captures the pair-wise relationship between two sets of features but also encodes the difference between each pair of features. It is likely that the pair-wise difference leads to better discriminant ability than that of the outer product. Our bilinear CNN is referred to as the Pair-wise Difference Pooling Based Bilinear CNN (BCNN-PDP). The PDP can also be applied to two sets of feature maps extracted from two images, respectively, using the same CNN. Due to the importance of the gradient data to the description of image structures, we utilise the PDP along with the original image and its gradient magnitude map. We refer to this PDP-based BCNN as the Fused BCNN-PDP (F-BCNN-PDP). In addition, the PDP-based BCNN can be implemented using two different CNNs, namely, the Asymmetric BCNN-PDP (A-BCNN-PDP).

The PDP-based BCNNs can also be computed using multiple scales. The three multi-scale PDP-based BCNNs are named MS-BCNN-PDP, MS-F-BCNN-PDP and MS-A-BCNN-PDP respectively. The BCNN-PDP is built on top of a Symmetric Pair-wise Difference Pooling function, while the F-BCNN-PDP and A-BCNN-PDP are implemented based on an Asymmetric Pair-wise Difference Pooling function.

Since BCNN feature vectors have a high dimensionality (e.g. 512×512), we further introduce a simple Block-wise PCA (BPCA) method, to generate more compact feature vectors.

Figure 1 shows the pipeline of the BCNN-PDP and F-BCNN-PDP. Similarly, the A-BCNN-PDP is implemented using two different CNNs.

## Symmetric Pair-wise Difference Pooling (SPDP)

Given an image (, ), the feature maps computed using a CNN can be written as an array in which is an feature map. At a location  (, ) in these feature maps, we have a dimensional feature vector In terms of this vector, the symmetric pair-wise difference (SPD)  can be computed as follows:

|  |  |
| --- | --- |
| . | (1) |

Furthermore, can be written as

|  |  |
| --- | --- |
| . | (2) |



Fig. 1. The flowchart of the proposed BCNN-PDP (indicated by blue) and F-BCNN-PDP (indicated by red) along with the Block-wise PCA (BPCA). Here, the VGG-VD-16 [44] is used in Streams A and B. For the asymmetric BCNN-PDP (A-BCNN-PDP), the CNN shown in Stream B is replaced by a different CNN while the other modules are the same as those displayed for the BCNN-PDP.

Across all the locations , the symmetric pair-wise difference pooling (SPDP) is calculated by applying the average pooling to the corresponding

|  |  |
| --- | --- |
| . | (3) |

If we substitute Equation (2) into Equation (3), then we have

|  |  |
| --- | --- |
| , | (4) |
| , | (5) |
| , | (6) |
| , | (7) |
| , | (8) |
| , | (9) |
| , | (10) |

where is the sum of the elements contained in the feature map . Therefore, can be easily computed.

The SPDP feature vector is further processed using the power law normalisation as:

|  |  |
| --- | --- |
| , | (11) |

where is the sign function and is the power law coefficient. Finally, the normalisation is applied to and we have

|  |  |
| --- | --- |
| . | (12) |

The feature vector is used as the representation of image .

## Asymmetric Pair-wise Difference Pooling (APDP)

Two sets of feature maps either can be computed from two images and (, ), respectively, using the same CNN, or can be computed from the same image using two different CNNs respectively. The two sets of feature maps are denoted as two arrays: and , respectively. In these arrays or is an feature map. Regarding a location (, ), we have two dimensional feature vectors: and . Using both the vectors, the asymmetric pair-wise difference (APD) can be calculated as

|  |  |
| --- | --- |
| . | (13) |

Similar to the deduction procedure introduced in Section III-A, the asymmetric pair-wise difference pooling (APDP) can be computed as

|  |  |
| --- | --- |
| . | (14) |

where and are the sums of the elements contained in the feature maps and respectively. Again, Equation (14) can be conveniently calculated.

The feature vector is further processed by using the power law and normalisations as shown in Equations (11) and (12) respectively.

## Pair-wise Difference Pooling Based BCNN (BCNN-PDP)

The SPDP is used to replace the outer product pooling of the BCNN proposed by Lin *et al*. [36]. We refer to this feature set as the Pair-wise Difference Pooling Based BCNN (BCNN-PDP). It can be directly computed from the feature maps extracted from an image using a pre-trained CNN.

## Fused Pair-wise Difference Pooling Based BCNN (F-BCNN-PDP)

It has been shown that the contour data is key to human perception and can be used for image representation [19-20], [22]. However, these data cannot be accurately obtained from some images. In contrast, the gradient data can be easily computed from an image and is also able to retain the image structure [24]. Therefore, the BCNN-PDP is further extended to two sets of feature maps calculated from the original image and the corresponding gradient magnitude map respectively. Since this method fuses the two sets of feature maps into a single vector, we term it the Fused BCNN-PDP (F-BCNN-PDP).

When a grey level image is processed, two derivative maps: and are calculated using the Canny [6] detector at the and directions respectively. The gradient magnitude map is computed according to

|  |  |
| --- | --- |
| . | (15) |

The computation process for colour images is more complicated. Given an RGB image, a pair of and are first computed from each colour channel using the Canny [6] detector at the and directions respectively. In total, six derivative maps: , , , , and are derived. The Jacobi eigenvalue algorithm [12] is used to compute the gradient map from these derivative maps. Let the Jacobi matrix be expressed as

|  |  |
| --- | --- |
| , | (16) |

we have

|  |  |
| --- | --- |
|  | (17) |
|  | (18) |
| . | (19) |

Thus, the Jacobian determinant is calculated as

|  |  |
| --- | --- |
| . | (20) |

The gradient magnitude can be calculated as the greatest eigenvalue of Equation (19) according to

|  |  |
| --- | --- |
| . | (21) |

Regarding image , F-BCNN-PDP features are computed from the two sets of feature maps extracted from this image and the corresponding using Equation (14). The features encode both the original image characteristics and the gradient information.

## Asymmetric Pair-wise Difference Pooling Based BCNN (A-BCNN-PDP)

Lin *et al.* [36] applied the outer product based BCNN to the feature maps calculated using two different CNNs. Compared with the single network BCNN feature set, the dual-network BCNN feature set produced better results. We are motivated to adapt the asymmetric BCNN-PDP (A-BCNN-PDP) by fusing the two sets of feature maps calculated from the same image using two different CNNs: VGG-M [9] and VGG-VD-16 [44].

## A Multi-scale Scheme for the PDP-Based BCNNs

Considering the features extracted using multiple scales normally produce better results than those computed at a single scale [11], we extend the PDP-based BCNNs to the multi-scale cases. In total six different scales: ( {-2, -1.5, -1, -0.5, 0, 0.5}) are used for an image. The multi-scale PDP-based BCNN is implemented using the Average Scale-wise Pooling. Given an image , specifically, six scale images are obtained at each scale using the cubic interpolation algorithm. For each scale image , a PDP-based BCNN feature vector is first computed separately and is denoted as . The multi-scale PDP-based BCNN feature vector is then calculated according to

|  |  |
| --- | --- |
| , | (22) |

where denotes the cardinality of .

In terms of the BCNN-PDP, F-BCNN-PDP and A-BCNN-PDP feature sets, the multi-scale feature sets are referred to as the MS-BCNN-PDP, MS-F-BCNN-PDP and MS-A-BCNN-PDP respectively.

## Block-wise PCA

Although the performance of the original BCNN [36] method is promising, the high dimensionality of the feature vectors that it produces limits its application to large scale datasets and also decreases the computational speed of recognition. This is also the case for the proposed PDP-based BCNN feature sets. In [36], Lin *et al*. used PCA reduce the dimensionality of one of the two sets of feature maps. The BCNN was computed from the reduced feature maps and non-reduced feature maps. However, this strategy impaired the recognition performance.

In essence, each element of the BCNN feature vector encodes the relationship between a feature map and another feature map (see Equations (5) and (6) for more details). Therefore, different elements can be treated as individual features. In this context, a PCA operation can be applied to a subset of the feature vector. Based on this assumption, we introduce a simple Block-wise PCA (BPCA) method. Given a set of BCNN features, in essence they are a matrix which can be expressed as

|  |  |
| --- | --- |
| . | (23) |

Given the matrix is divided into a set of blocks, (), we first apply an individual PCA operation to each block as described below:

|  |  |
| --- | --- |
| . | (24) |

The PCA operation comprises four procedures, including an normalisation, PCA, whitening and a second normalisation. Then, all are collapsed into a single feature vector based on the concatenation operation:

|  |  |
| --- | --- |
| . | (25) |

Compared with the original feature vector, the feature vector is more compact. (Given the block size is () and the output dimensionality of each PCA is (), the dimensionality of the feature vector is ()). As a result, the computational speed of image classification is enhanced.

# Experimental Setup

In this section, we introduce the experimental setup. First, we briefly present the datasets, classifier and performance measure used in our experiments. Second, we describe the baseline feature sets which are used for comparison purposes. Third, we present the implementation details of our experiments.

## Datasets, Classifier and Performance Measure

We test the proposed PDP-based BCNN feature sets using seven publicly available datasets, including six texture datasets and an aerial scene dataset which contains some textural classes. Three texture datasets contain colour images while the other three texture datasets comprise grey level images. For the aerial scene dataset, colour images are included.

A Support Vector Machine (SVM) [26] classifier is used for the classification task. For each class of the six texture datasets, 50% images are randomly selected for training and the other 50% images are used for testing. In total, ten random splits are generated for each texture dataset. Mean and standard deviation are computed across the ten classification accuracy (%) values obtained using each split, and are used as the performance measure. With regard to the aerial scene dataset, a five-fold cross validation experiment is performed by following the original experimental setup in [53]. The average classification accuracy (%) calculated across the five folds is used as the performance measure.

## Baseline Feature Sets

In order to fairly evaluate the proposed feature sets, we use six hand-crafted texture feature sets, four types of deep visual word feature sets, one deep fully-connected feature set and the outer product based BCNN [36] feature sets as the baselines.

Two different normalisation strategies are used for the hand-crafted and CNN feature sets, respectively, before features are extracted. Regarding the hand-crafted feature sets, we normalise each image in order to ensure that this image has zero mean and unit standard deviation by following the setup used by Ojala *et al*. [42]. This operation removes the impact of 1st- and 2nd-order grey level statistics on the image. For the CNN feature sets, we follow the setup that Cimpoi *et al*. [11] used in which each image is subtracted by the average colour of the dataset used for training the CNN.

### Hand-Crafted Texture Feature Sets

**Gabor Wavelet Filter Bank (GaborMM) [39]** The magnitude maps of the complex filtered images calculated at six orientations and four scales are computed. For each magnitude map, mean and standard deviation are calculated. All the means and standard deviations are concatenated into a 48-D feature vector.

**Grey Level Co-occurrence Matrices (GLCM) [29]** The input image is quantised to 32 grey levels. In total, 16 displacements are used, including four distance values and four directions. In terms of the co-occurrence matrix obtained using each displacement, 22 statistics are computed. Given a distance value, mean and standard deviation are computed across different directions for each statistic. All the means and standard deviations are combined into a 176-D feature vector.

**Local Binary Patterns (LBP), Local Variances (VAR) and Their Joint Distribution (LBP/VAR)** The “uniform” and the grey-scale and rotation invariant version: [42] is used for the LBP feature set. The rotation invariant variance measure (i.e. VAR) and the joint distribution of both the feature sets: (i.e. LBP/VAR) proposed in [42] are also utilised. We set to 16 for the three feature sets as this value performed the best in [42].

**Multi-resolution Simultaneous Autoregressive Models (MRSAR)** Three resolutions are used for the MRSAR [40] feature sets. With regard to a local region at each resolution, four coefficients and the standard deviation of the estimation errors are computed using the least-squares algorithm. As a result, five feature matrices are produced. The means and standard deviations are computed from each feature matrix and are combined into a 30-D feature vector.

### Deep Visual Word Feature Sets

We extract multi-scale deep convolutional features at the last convolutional layer of the pre-trained VGG-VD-16 [44] model. In total, six scales are used as we perform for the multi-scale PDP-based BCNN feature sets (see Section III-F). Deep visual words are learnt from the multi-scale deep convolutional features using *k*-means or Gaussian Mixture Model (GMM). For more details, please refer to [11] and [22]. We learn three sets of words, including 50, 100 and 200 words, respectively. Four methods are used to encode the deep words as introduced below. In total, 12 sets of deep word features are obtained.

**Bag-of-Words (BoW)** The BoW [45] feature set is generated via accumulating a histogram from the occurrence frequency of the label of each visual word.

**Fisher Vectors (FV)** The GMM is used for the FV [31] feature set. We calculate FV features using the signed square-rooting and normalisation.

**Spatial Pyramid Matching (SPM)** Weextract a BoW [45] feature vector at each of three levels of spatial pyramids. These feature vectors are weighted and concatenated using the method introduced by Lazebnik *et al*. [33].

**Vector of Locally Aggregated Descriptors (VLAD)** The signed square-rooting and global normalisation approaches are used for the VLAD [30] feature set.

### The Deep Fully-Connected (FC) Feature Set

Following the work of Cimpoi *et al*. [11], we extract the 4096-D FC features at the penultimate fully-connected layer of the pre-trained VGG-VD-16 [44] model.

### BCNN

The BCNN [36] feature set is extracted from the last convolutional layer of the pre-trained VGG-VD-16 [44] model. The proposed block-wise PCA can also be applied to this feature set.

## Implementation Details

We extract the BCNN-PDP, F-BCNN-PDP and BCNN [36] features from the last convolutional layer of the pre-trained VGG-VD-16 [44] model. The A-BCNN-PDP features are computed from the last convolutional layers of both the pre-trained VGG-VD-16 [44] and VGG-M [9] models. For the F-BCNN-PDP feature set, the sigma used for the *Canny* detector [6] is set to 1.0. The value of used for the power law is set to 0.5.

For the *VisTex* [1] dataset, all the images are used to learn the block-wise PCA projections because this dataset is relatively small. In contrast, 33% images are employed for the other datasets. The size of the blocks is set to 2048 and the dimensionality of the output of each PCA is set to 128. Therefore, for a 512×512 feature vector, four columns are used for learning a single PCA. The reduced dimensionality of the feature vector is 128×128=16,384. The linear SVM [26] is applied with the value of set to 10.

# Experiments

The three proposed pair-wise difference pooling (PDP) based BCNN feature sets and their multi-scale versions are tested on the six texture datasets and an aerial scene dataset along with 21 baseline feature sets. The setup introduced in Section IV is used. For the computational efficiency purpose, we mainly use the PDP-based BCNN feature sets together with the proposed block-wise PCA. We report the experimental results in detail as follows.

## Brodatz

As one of the most popular texture datasets in the computer vision community, *Brodatz* consists of 112 texture images taken from a photo album [5]. Each image was treated as a single class and was divided into 64 80×80 non-overlapping patches. As a result, 7,168 image patches were obtained. Figure 2 shows the first 21 *Brodatz* textures.

The classification results are displayed in the second column of Table I. As can be seen, (1) the LBP/VAR [42] and MRSAR [40] feature sets outperformed the other hand-crafted feature sets; (2) the results of the visual word feature sets were normally superior to those of the hand-crafted feature sets; (3) VGG-VD-16 [44] outperformed all the visual word feature sets and BCNN [36]; (4) BCNN [36] performed better than the hand-crafted feature sets and the visual word feature sets; (5) the use of the proposed block-wise PCA improved the performance of BCNN; (5) the proposed BCNN-PDP feature set outperformed BCNN. In particular, all the proposed feature sets produced better results than their counterparts; (6) both the F-BCNN-PDP-BPCA and A-BCNN-PDP-BPCA feature sets performed better than, or comparably to BCNN-PDP; (7) our multi-scale scheme improved the results of BCNN-PDP, F-BCNN-PDP-BPCA and A-BCNN-PDP-BPCA; and (8) the best result: 97.57±0.24 was achieved by our multi-scale A-BCNN-PDP-BPCA.

## CUReT

The *CUReT* dataset [16] was acquired in order to capture the visual appearance of real-world surfaces. In total, 61 texture samples were included in this dataset. Each sample was regarded as an individual class and over 200 images were acquired under different viewing and illumination directions from this sample. We used the subset that Varma and Zisserman [48] utilised. For each class, 92 200×200 images were derived. As a result, 5,612 images were used. Figure 3 displays the examples of the first 21 classes.

The third column of Table I shows the results obtained using the *CUReT* dataset [16]. It can be observed that (1) both MRSAR [40] and GaborMM [39] outperformed the other hand-crafted feature sets with large margins. They were also superior to the BoW feature sets; (2) the FV [31] and VLAD [30] feature sets performed better than the hand-crafted, BoW [45], SPM [33], VGG-VD-16 [44] and BCNN [36] feature sets; (3) BCNN produced the results similar to that generated by the hand-crafted feature set: MRSAR [40]. But they were outperformed by VGG-VD-16 and our feature sets; (4) the block-wise PCA improved the performance of both BCNN [36] and BCNN-PDP; (5) the proposed PDP-based BCNN feature sets together with the block-wise PCA were superior to their counterparts; (6) F-BCNN-PDP and A-BCNN-PDP performed slightly better than the BCNN-PDP. The multi-scale scheme did boost the performance of these feature sets; and (7) the best performance was provided by the proposed multi-scale F-BCNN-PDP feature set along with the block-wise PCA.

## DTD

TABLE I

The average classification accuracy (%) values obtained using the four groups of baseline feature sets and the proposed PDP-based BCNN feature sets on six texture datasets and an aerial scene dataset. For the proposed feature sets and the BCNN [36] feature set, “-BPCA” means that the block-wise PCA has been applied. For each texture dataset, the mean and standard deviation of the accuracy values are computed across ten random splits. Regarding the aerial scene dataset, a five-fold cross validation scheme is used and the average classification accuracy is calculated across the five folds, following the experimental setup in [53].

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***Brodatz*** | ***CUReT*** | ***DTD*** | ***Pertex*** | ***sTex*** | ***VisTex*** | ***Aerial Scene*** |
| **GaborMM** | 73.37±0.35 | 95.91±0.27 | 18.61±0.58 | 59.30±0.48 | 55.69±0.64 | 62.59±1.93 | 52.38 |
| **GLCM** | 58.08±0.56 | 83.54±0.61 | 12.56±0.48 | 14.59±0.58 | 46.34±0.92 | 53.93±0.85 | 40.86 |
| **LBP** | 69.65±0.34 | 55.82±0.63 | 15.70±0.57 | 80.33±0.76 | 19.88±0.84 | 36.20±1.74 | 40.43 |
| **VAR** | 33.41±0.44 | 30.31±0.63 | 8.36±0.43 | 50.63±0.85 | 16.13±0.98 | 28.29±1.49 | 36.05 |
| **LBP/VAR** | 75.40±0.75 | 64.29±0.70 | 17.88±0.46 | 87.56±0.59 | 40.26±1.12 | 52.34±1.59 | 53.67 |
| **MRSAR** | 75.00±0.24 | 97.62±0.26 | 18.17±0.70 | 43.52±0.68 | 82.12±0.99 | 83.61±0.84 | 57.33 |
| **BoW-W50** | 73.55±0.24 | 89.13±0.54 | 53.14±0.98 | 77.43±0.81 | 65.19±0.91 | 66.13±1.99 | 90.10 |
| **BoW-W100** | 84.11±0.42 | 92.83±0.45 | 57.14±0.97 | 84.71±0.75 | 78.45±0.93 | 77.73±1.09 | 91.86 |
| **BoW-W200** | 88.53±0.48 | 95.27±0.31 | 59.73±0.80 | 88.27±0.83 | 86.03±1.31 | 81.00±1.52 | 93.95 |
| **SPM-W50** | 68.76±0.73 | 92.27±0.39 | 52.94±0.90 | 76.89±0.69 | 62.73±1.25 | 66.63±1.41 | 89.67 |
| **SPM-W100** | 81.63±0.37 | 94.67±0.31 | 57.01±0.89 | 83.90±0.54 | 75.52±1.29 | 77.38±1.54 | 92.48 |
| **SPM-W200** | 86.77±0.67 | 96.36±0.30 | 59.70±0.90 | 87.85±0.57 | 84.11±1.27 | 80.34±1.56 | 94.43 |
| **FV-W50** | 95.20±0.43 | 99.39±0.21 | **71.53±0.77** | 98.64±0.15 | 87.50±2.04 | 75.09±1.30 | 96.19 |
| **FV-W100** | 94.45±0.60 | 99.39±0.18 | 71.41±0.95 | 98.50±0.16 | 87.09±0.72 | 67.68±2.19 | 95.38 |
| **FV-W200** | 92.50±0.48 | 99.06±0.22 | 71.13±0.82 | 98.39±0.21 | 80.96±1.09 | 54.09±3.50 | 94.90 |
| **VLAD-W50** | 94.80±0.32 | 99.18±0.19 | 70.23±0.84 | 98.62±0.18 | 90.96±3.06 | 76.38±2.18 | 96.10 |
| **VLAD-W100** | 92.01±0.32 | 99.19±0.25 | 71.02±0.77 | 98.62±0.20 | 87.63±1.08 | 68.55±2.06 | 95.81 |
| **VLAD-W200** | 90.15±0.57 | 98.97±0.22 | 70.70±0.73 | 98.45±0.23 | 83.38±1.34 | 60.61±2.88 | 94.43 |
| **VGG-VD-16** | 95.74±0.27 | 98.47±0.29 | 64.94±0.78 | 90.86±0.61 | 96.58±0.43 | **87.21±0.75** | 92.76 |
| **BCNN** | 95.07±0.35 | 97.77±0.31 | 66.11±0.88 | 88.76±0.56 | 95.00±0.55 | 79.55±1.09 | 94.24 |
| **BCNN-BPCA** | 95.66±0.29 | 99.33±0.20 | 66.24±0.76 | 97.22±0.33 | 97.95±0.36 | 84.09±0.97 | 94.86 |
| **BCNN-PDP** | 96.48±0.26 | 99.07±0.10 | 68.46±0.92 | 96.42±0.25 | 98.13±0.30 | 86.20±1.37 | 95.67 |
| **BCNN-PDP-BPCA** | 96.41±0.25 | 99.40±0.22 | 66.76±0.86 | 98.16±0.30 | 98.23±0.45 | 85.27±1.01 | 95.81 |
| **MS-BCNN-PDP-BPCA** | 97.00±0.35 | 99.68±0.08 | 69.67±0.95 | 98.17±0.12 | 98.56±0.33 | 86.43±1.21 | 96.10 |
| **F-BCNN-PDP-BPCA** | 96.36±0.26 | 99.46±0.14 | 67.02±0.89 | 98.69±0.19 | 98.48±0.40 | 85.66±1.34 | **96.48** |
| **MS-F-BCNN-PDP-BPCA** | 97.23±0.21 | **99.73±0.08** | 70.02±0.88 | 98.76±0.16 | **98.77±0.24** | 86.71±1.32 | 96.43 |
| **A-BCNN-PDP-BPCA** | 96.91±0.23 | 99.55±0.13 | 66.96±0.86 | **98.80±0.19** | 98.57±0.39 | 86.14±1.44 | 96.33 |
| **MS-A-BCNN-PDP-BPCA** | **97.57±0.24** | 99.71±0.07 | 69.95±1.01 | 98.65±0.22 | 98.73±0.25 | 87.05±1.29 | **96.48** |

The *Describable Textures Dataset* *(DTD)* [11] was used to identifying describable texture attributes in images. In total, 47 texture attributes were selected. Each one was used as the key attribute of 120 texture images. As a result, 5,640 images were included in this dataset. The key attributes were treated as classes. The examples of the first 21 classes are shown in Fig. 4.

It can be seen from the fourth column of Table I that (1) the hand-crafted feature sets performed poorly and were inferior to all the CNN-based feature sets; (2) both FV [31] and VLAD [30] outperformed BoW [45] and SPM [33] with large margins. They were also superior to VGG-VD-16 [44], BCNN [36] and the proposed feature sets. However, our best performance 70.02%±0.88 is close to the results produced by FV [31] and VLAD [30]; (3) the proposed feature sets performed better than their counterparts excepting FV and VLAD; (4) both F-BCNN-PDP and A-BCNN-PDP yielded better results than BCNN-PDP; and (5) the multi-scale scheme indeed improved the performance of the proposed feature sets.

In [51], the performances obtained using the end-to-end trained Deep-TEN [56] and DEP [51] were 69.6% and 73.2% respectively. Compared with these values, our best result 70.02%±0.88 is close even if we used a pre-trained CNN. Furthermore, the results produced by the compact BCNN [27] and the Locally-Transferred Fisher Vectors (LFV) method [46] reported by Song et al. [46] were 67.71% and 68.6%±1.0 respectively. Although these results are worse than the best performance of our methods, it should be noted that they used one third of the images in each class for training and the rest for testing.

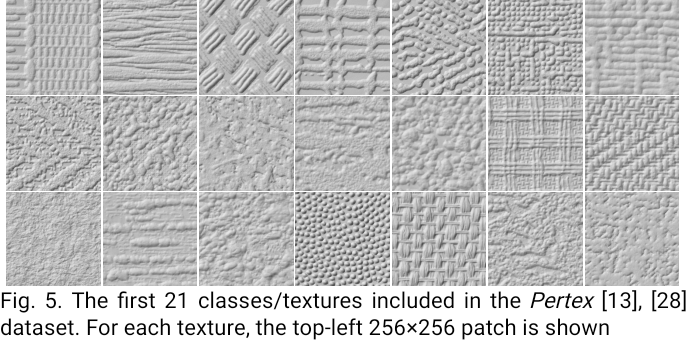
## Pertex

The *Pertex* [13], [28] dataset was collected for acquiring human perceptual texture similarity data. In total, 334 textures are included in this dataset. The dataset also provides the height map for each texture. We used the images published by Clarke *et al*. [13], which had been rendered using the same lighting condition. We divided each 1024×1024 texture image into 16 256×256 non-overlapping patches. Hence, 5,344 patches were produced. Each texture was considered as a single class. Figure 5 shows the first 21 classes of the *Pertex* dataset.

The results are reported in the fifth column of Table I. It is shown that (1) the LBP/VAR [42] feature set outperformed the other hand-crafted feature sets and produced results similar to those obtained using the BoW [45], SPM [33] and BCNN [36] feature sets. However, it was inferior to the VGG-VD-16 [44], FV [31] and VLAD [30] feature sets and our PDP-based BCNN feature sets; (2) the use of the block-wise PCA boosted the performance of BCNN [36] greatly; (3) the proposed BCNN-PDP feature set performed much better than the BCNN [36] feature set; (4) both the F-BCNN-PDP-BPCA and A-BCNN-PDP-BPCA feature sets produced better results than that generated by the BCNN-PDP feature set; and (5) the best result was derived using our A-BCNN-PDP-BPCA feature set.

## sTex

The *sTex* [2] dataset consists of 31 categories of textures. To tackle the class imbalance issue, only the first eight textures in each category were used and the categories which contain less than eight textures were left out. As a result, 20×8=160 textures were selected. Figure 6 presents the examples of the 20 categories. Each texture image was regarded as a single class and was divided into 16 128×128 non-overlapping patches. In total, 160×16=2,560 patches were obtained.

The six column of Table I displays the results obtained using the *sTex* dataset. We can observe that (1) among the hand-crafted feature sets, MRSAR [40] yielded the best result, which was much better than those produced by the other hand-crafted feature sets and was better than, or comparable to, those obtained using BoW [45] and SPM [33]; (2) both FV [31] and VLAD [30] outperformed the BoW [45] and SPM [33] feature sets when 50 or 100 words were used, while it was not the case when 200 words were utilised; (3) VGG-VD-16 [44], BCNN [36] and the proposed feature sets were superior to their counterparts. In particular, VGG-VD-16 outperformed BCNN but it was surpassed by the BCNN feature set when the block-wise PCA was applied to the latter; (4) all the proposed feature sets performed better than the other feature sets; (5) both F-BCNN-PDP-BPCA and A-BCNN-PDP-BPCA were superior to BCNN-PDP-BPCA. The multi-scale scheme further improved the performance of these feature sets; and (6) the best result was provided by the multi-scale F-BCNN-PDP-BPCA.

## VisTex

The *VisTex* [1] dataset comprises 19 categories of textures. The categories which contain less than five textures were ignored and only the first five textures of each category were selected in order to avoid the class imbalance problem. Finally, 14×5=70 textures were selected. Figure 7 displays the examples of the 14 categories. Each texture was treated as a single class and was divided into 16 128×128 non-overlapping patches. As a result, 70×16=1120 patches were obtained in total.

From the results shown in the seventh column of Table I, we can see that (1) the MRSAR [40] feature set outperformed the other hand-crafted feature sets with large margins. It was also superior to some CNN-based feature sets, including BoW [45], SPM [33], FV [31], VLAD [30] and BCNN [36]; (2) BCNN yielded better results than those obtained using the hand-crafted feature sets excepting MRSAR [40]. It also outperformed BoW [45], SPM [33], FV [31] and VLAD [30] with the exception that it was inferior to BoW and SPM when 200 words were used; (3) our block-wise PCA improved the performance of BCNN; (4) the proposed feature sets outperformed all their counterparts except the VGG-VD-16 [44] feature set; (5) the multi-scale scheme worked for the proposed feature sets; and (6) the best result 87.21±0.75 was provided by VGG-VD-16 [44], which was very close to the accuracy of 87.05±1.29 produced by the proposed multi-scale A-BCNN-PDP-BPCA feature set.

## Aerial Scene

To further augment the experimental results, we tested the proposed feature sets and the baseline methods on an aerial scene dataset [53]. This dataset includes 21 classes and 100 images are contained in each class. Different objects and spatial patterns are presented in these images. The examples of each class are shown in Fig. 8. Following the experimental setup used by Yang and Newsam [53], the five-fold cross-validation classification was run. The average classification accuracy (%) was computed across the five folds.

As can be seen in the eighth column of Table I, (1) all the hand-crafted feature sets performed significantly worse than the CNN-based feature sets. The best result produced by these feature sets was only 57.33%; (2) both FV [31] and VLAD [30] outperformed BoW [45], SPM [33], VGG-VD-16 [44] and BCNN [36]; (3) benefiting from the proposed block-wise PCA, the performance of BCNN [36] was boosted and was superior to BoW [45], SPM [33] and VGG-VD-16 [44]; (4) the proposed BCNN-PDP outperformed BCNN [36] but it was inferior to both F-BCNN-PDP-BPCA and A-BCNN-PDP-BPCA; (5) the best result was provided by the proposed F-BCNN-PDP-BPCA and MS-A-BCNN-PDP-BPCA.

## Summary

In this section, the proposed feature sets have been tested on seven different datasets along with 21 baseline approaches. Figure 9 displays the results produced by the proposed BCNN feature sets and three baselines: VGG-VD-16 [44], BCNN [36] and BCNN-BPCA. According to the results shown in Fig. 9 and Table I, it can be found that (1) the CNN-based feature sets normally produced better results than those obtained using the hand-crafted feature sets. Although hand-crafted feature sets performed properly on some datasets, their performance varied much across the datasets. In contrast, the learning-based CNN feature sets were more generic than the hand-crafted feature sets. This finding is consistent with that reported by Cimpoi *et al*. [11]; (2) the performance of BCNN [36] was usually inferior to that of VGG-VD-16 [44] even though it performed better for the fine-grained image recognition task [36]; (3) the proposed block-wise PCA was able to boost the performance of BCNN and the proposed feature sets usually. In particular, it was effective for BCNN. The effectiveness of our method should be due to the fact that it reduces the dimensionality of each subset of the BCNN features while retaining the rough pair-wise relationship encoded by these features; (4) the proposed multi-scale scheme normally enhanced the performance of our feature sets. This observation is consistent with that demonstrated by Cimpoi *et al*. [11]; (5) both the F-BCNN-PDP and A-BCNN-PDP feature sets normally performed better than the BCNN-PDP method because they encoded richer image characteristics; and (6) the F-BCNN-PDP and A-BCNN-PDP feature sets provided the best results on five of the seven datasets among all the feature sets examined here.

# Detailed Examination of the PDP-Based BCNNs

To augment the experimental results, we further examine the proposed PDP-based BCNN feature sets in more detail. Specifically, we change the modules used for the classification task and tune the parameters of these feature sets. For simplicity, we only use the *sTex* [2] dataset.

## Comparison with Random Forests

We compared the Random Forest (RF) [4] classifier with the SVM [26] used in Section V. Following the setup that Breiman [4] used, we randomly selected a subset of features for the -dimensional feature vector. The minimum size of the terminal nodes was set to 0.01% of the number of the training samples. For each branch node of the subset, we utilised the *Gini* impurity measure [4] for the feature and decision boundary selections. Table II reports the classification accuracy (%) values obtained using the SVM and RF classifiers along with the BCNN-PDP and BCNN [36] feature sets. It can be seen that (1) SVM was superior to RF no matter which bilinear CNN feature set was used; and (2) BCNN-PDP always outperformed BCNN.

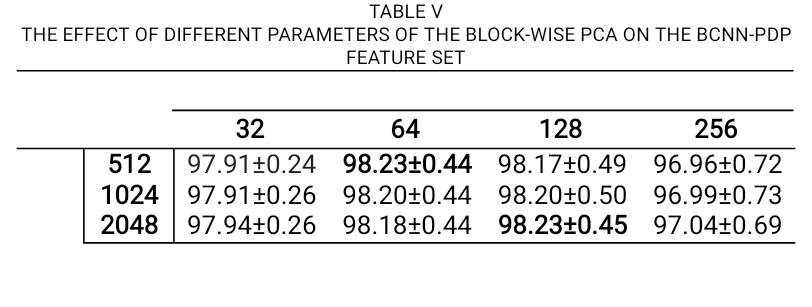
## Comparison of Different CNNs

Different pre-trained CNNs, including AlextNet [32] and VGG-M [9] were also used to compute the BCNN-PDP and BCNN [36] feature sets. The results are compared with those produced by the VGG-VD-16 [44] CNN in Table III**.** It can be seen that VGG-VD-16 usually generated better results than those produced by the AlextNet [32] and VGG-M [9] CNNs for both the BCNN-PDP and BCNN feature sets. Again, our BCNN-PDP feature set was always superior to BCNN. However, the results: 97.59±0.44 and 98.23±0.45 obtained using BCNN-PDP-BPCA with the VGG-M and VGG-VD-16 CNNs, respectively, were worse than the result: 98.57±0.39 produced by the proposed A-BCNN-PDP-BPCA which utilised both the CNNs.

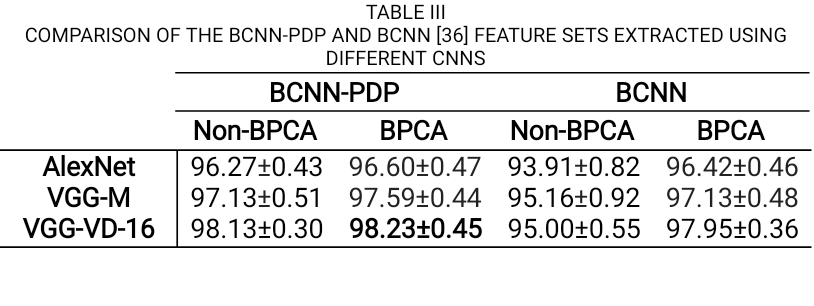
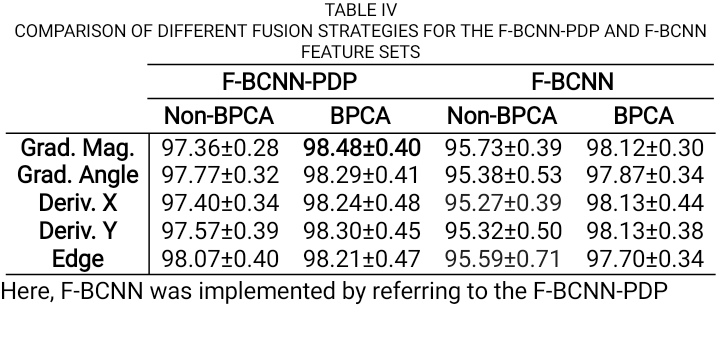
## Comparison of Different Fusion Strategies

Our F-BCNN-PDP feature set fuses the features extracted from the original image and the associated gradient magnitude map. To compare different fusion strategies, we replaced the magnitude map by different types of images calculated from the original image, including the gradient angle map, the directional derivative maps in the and directions and the edge map. We also implemented the fused BCNN (F-BCNN) for the comparison purpose. The results are shown in Table IV. As can be seen, the F-BCNN-PDP together with the block-wise PCA achieved its best performance 98.48±0.40 when the gradient magnitude map was used. Again, the F-BCNN-PDP feature set always outperformed F-BCNN.

## Effect of the Parameters of the Block-wise PCA

The proposed block-wise PCA has two parameters: the block size, , and the output dimensionality of each PCA, . To explore the effect of the two parameters on the classification task, we set these to different values. The results obtained using different combinations of and are reported in Table V. It can be seen that the best performance was derived when and were set to 512 and 64 or 2048 and 128. However, the latter combination produces fewer features and thus requires less memory. In fact, we have used this combination in Section V.

## Effect of Different Convolutional Layers

In this paper, we used the “Conv5\_3” layer of the VGG-VD-16 [44] CNN to extract the proposed BCNN-PDP features. We examined the effect of different convolutional layers on this feature set. Table VI reports the results produced by different convolutional layers. As can be seen, “Conv5\_1” generated the better result than the other layers while “Conv5\_3” performed comparably to it.

## Effect of the Coefficient of the Power Law

The proposed BCNN-PDP feature sets apply the power law to the feature vector after the pooling operation is complete. We tuned the value of the power coefficient . To investigate the best performance, the “Conv5\_1” layer was used. Table VII presents the classification accuracy values obtained using different values. The best result was produced when was set to 0.25. However, the default value used in this paper provided the close result.

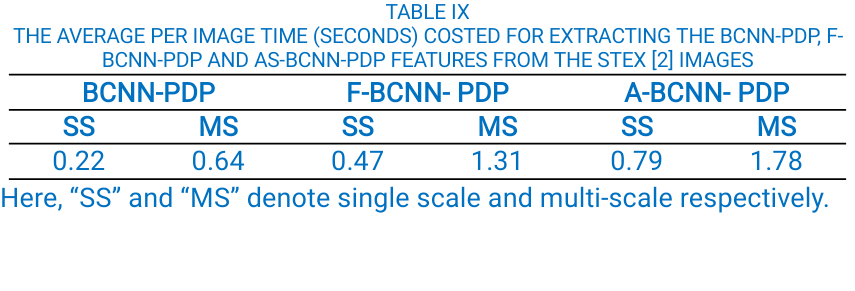
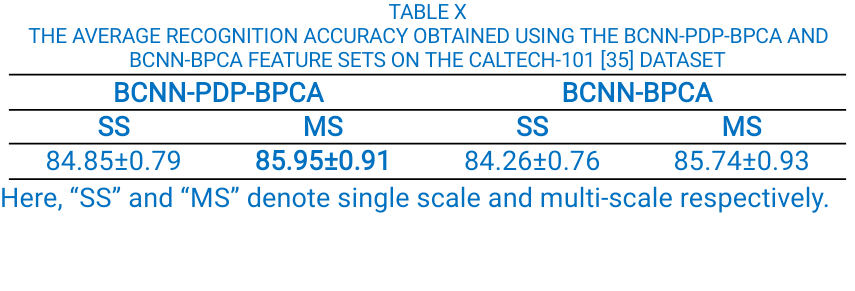
## Examples of Misclassified Images

With regard to the best result (i.e. 98.80%±0.33) presented in Section VI-F, we investigate the misclassified images produced in this case. In total, 154 images were misclassified across the ten splits of the experiment. Given that only the images which were misclassified at least twice across these splits were considered, 32 images were obtained. These images accounts for 71 misclassified cases. Fig. 10 shows the 32 images along with the ground-truth and classified class labels. It can be observed that 24 images were classified into the class which belongs to the same category (e.g. different flowers or foods) as that of the ground-truth class.

## Effect of the Number of Training Images

We examined the effect of the number of the training images on the classification performance. For simplicity, the BCNN-PDP-BPCA feature set extracted at the “Conv5\_1” layer was utilised. As shown in Table VIII, more training images produced better and more stable results.

## Computational Cost

Given that the experiment was performed on a Lenovo T540p laptop, which has a 64-bit, 2.5GHz Intel(R) i7-4710MQ CPU and 16.0 GB memory, Table IX reports the average per image time cost for extracting the BCNN-PDP, F-BCNN-PDP and A-BCNN-PDP features from the *sTex* [2] images using the single scale and multiple scales. It is shown that the computational speed for extracting the three feature sets is fast even if the feature extraction is only implemented on the CPU.

# Generalisation to Object Recognition

In this section, we applied the proposed method to the Caltech-101 [35] dataset for object recognition. This dataset contains 9,144 images, which were divided into 102 classes, including 101 object classes and one background class. Following the existing studies [9], [44], we randomly selected 30 images as the training images and up to 50 images as the test images for each class. In total, we generated ten splits. The average recognition accuracy (%) was computed across these splits as the mean class accuracy.

Table X reports the average recognition accuracy produced by the BCNN-PDP-BPCA and BCNN-BPCA feature sets using the single scale and multiple scales. As can be seen, the proposed BCNN-PDP-BPCA feature set always outperformed the BCNN-BPCA feature set.

# Conclusions and Future Work

In this paper, we applied the Pair-wise Difference Pooling (PDP) to the Bilinear Convolutional Neural Network (BCNN). The new BCNN is referred to as the PDP-Based BCNN (BCNN-PDP). In contrast to the outer product used by the original BCNN [36] approach, the pair-wise difference encodes both the pair-wise relationship between two sets of features and the difference between each pair of features. Therefore, BCNN-PDP owns a stronger discriminant power than the original BCNN method. In particular, it can be computed from a convolutional layer of a pre-trained CNN. We also adapted the PDP-based BCNN by taking into account two sets of feature maps calculated from the original image and its gradient magnitude map respectively. The adapted feature set utilises the characteristics presented in both the original image and the gradient map, and is termed the Fused BCNN-PDP or F-BCNN-PDP for short. Besides, two different CNNs can be used to compute the PDP-based BCNN features. We refer to this dual-CNN style PDP-based BCNN as the Asymmetric BCNN-PDP (A-BCNN-PDP).

Considering the importance of multiple resolutions to texture representation, we proposed a multi-scale scheme for the three proposed PDP-based BCNNs. Using this scheme, multiple scales of images are used for computing the PDP-based BCNN features. However, the high dimensionality of the BCNN feature sets limits the practical application of them. To address this issue, we introduced a simple Block-wise PCA (BPCA) method. This method can be applied to both the proposed and original BCNN feature sets. The three proposed feature sets together with the multi-scale scheme and the BPCA method were tested on seven image datasets. Our results showed that both the F-BCNN-PDP and A-BCNN-PDP feature sets normally produced better results than those generated by the BCNN-PDP feature set. The use of the block-wise PCA not only reduced the dimensionality of the features extracted using these feature sets, but also improved their performances in some cases. In addition, the multi-scale scheme was able to boost the performance of those feature sets further. Nonetheless, the most important finding is that the proposed methods outperformed, or performed comparably to, the 21 baseline feature sets. Particularly, the BCNN-PDP always produced better results than those derived using the original BCNN [36] method. These promising results should be due to the stronger discriminant power of the pair-wise difference than that of the outer product.

Although the use of the pre-trained CNN has shown the promising generalisation ability in this study, it has been demonstrated in the literature that the end-to-end trained CNN outperforms a pre-trained CNN when the sufficient training data is available. In this situation, the end-to-end training makes the CNN better suit the specific data set than the pre-trained CNN. In our future work, we aim to end-to-end train CNNs using the Pair-wise Difference Pooling (PDP). However, this study has shown that the PDP together with pre-trained CNNs is particularly effective for texture classification with small datasets without fine-tuning the CNNs.

Acknowledgment

The authors would like to thank the anonymous reviewers for their comments and suggestions to improve this manuscript.

References

1. http://vismod.media.mit.edu/vismod/imagery/VisionTexture/, 1995.
2. http://www.wavelab.at/sources/STex/.
3. S. Basu, S. Mukhopadhyay, M. Karki, R. DiBiano, S. Ganguly, R. Nemani, and S. Gayaka, “Deep neural networks for texture classification-A theoretical analysis,” *Neural Networks*, vol. 97, pp. 173-182, 2018.
4. L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
5. P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. Dover Publications, 1966.
6. J. Canny, “A Computational Approach to Edge Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679-698, 1986.
7. J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, “Semantic segmentation with second-order pooling,” in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 430-443.
8. T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng and Y. Ma, “PCANet: A Simple Deep Learning Baseline for Image Classification?,” *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017-5032, 2015.
9. K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the Devil in the Details: Delving Deep into Convolutional Networks,” in *Proc. British Machine Vision Conference*, 2014.
10. B.B. Chaudhuri, N. Sarkar, P. Kundu, “Improved fractal geometry based texture segmentation technique,” *IEE Proc. of Computers and Digital Techniques*, vol. 140, pp. 233-241, 1993.
11. M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi, “Deep Filter Banks for Texture Recognition, Description, and Segmentation,” *Int’l J. Computer Vision*, vol. 118, no. 1, pp. 65-94, 2016.
12. C. Clapham and J. Nicholson, *The Concise Oxford Dictionary of Mathematics (4th Ed.)*, Oxford University Press, 2009.
13. A.D.F. Clarke, F. Halley, A.J. Newell, L.D. Griffin, and M.J. Chantler, “Perceptual Similarity: A Texture Challenge,” in *Proc. British Machine Vision Conference*, 2011.
14. D. Coppersmith, S. J. Hong, and J. R. M. Hosking, “Partitioning Nominal Attributes in Decision Trees,” *Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 197-217, 1999.
15. N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
16. K.J. Dana, B. Van Ginneken, S.K. Nayar, and J.J. Koenderink, “Reflectance and Texture of Real World Surfaces,” *ACM Trans. on Graphics*, vol. 18, no. 1, pp. 1-34, 1999.
17. P. Domingos and M. Pazzani, “On the optimality of the simple Bayesian classifier under zero-one loss,” *Machine Learning*, vol. 29, no. 2, pp. 103-137, 1997.
18. J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,” in *Proc. International Conference on Machine Learning, PMLR*, vol. 32, no. 1, pp. 647-655, 2014.
19. X. Dong, “Perceptual Texture Similarity Estimation,” PhD thesis, Heriot-Watt University, 2014.
20. X. Dong and M. J. Chantler, “Perceptually Motivated Image Features Using Contours,” *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5050-5062, 2016.
21. X. Dong and J. Dong, “Oceanic Scene Recognition Using Graph-of-Words (GoW),” in *Proc. IEEE International Conference on Computer Vision Workshops*, pp. 1122-1130, 2017.
22. X. Dong and J. Dong, “The Visual Word Booster: A Spatial Layout of Words Descriptor Exploiting Contour Cues”, *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3904-3917, 2018.
23. X. Dong, J. Dong, and M. J. Chantler, “Perceptual Texture Similarity Estimation: An Evaluation of Computational Features,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, DOI: 10.1109/TPAMI.2020.2964533, 2020.
24. X. Dong, J. Dong, S. Wang, and M.J. Chantler, “Perceptual Texture Retrieval Using Spatial Distributions of Textons (SDoT),” in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 663 - 666, 2015.
25. X. Dong, T. Methven, and M. J. Chantler, “How Well Do Computational Features Perceptually Rank Textures? A Comparative Evaluation,” in *Proc. the ACM 2014 International Conference on Multimedia Retrieval*, pp. 281-288, 2014.
26. R. Fan, P. Chen, and C. Lin, “Working set selection using the second order information for training SVM,” *Journal of Machine Learning Research*, vol. 6, pp. 1889-1918, 2005.
27. Y. Gao, O. Beijbom, N. Zhang and T. Darrell, “Compact Bilinear Pooling,” in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 317-326, 2016.
28. F. Halley, “*Pertex v1.0*,” 2011; http://www.macs.hw.ac.uk/ texturelab/resources/databases/pertex/.
29. R. Haralick, K. Shanmugam, and I. Dinstein, “Textural Features for Image Classification,” *IEEE Trans. Systems, Man, Cybernetics*, vol. 3, pp. 610-621, 1973.
30. H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, “Aggregating Local Image Descriptors into Compact Codes,” *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 34, no. 9, pp. 1704-1716, 2012.
31. J. Krapac, J. Verbeek, and F. Jurie, “Modeling spatial layout with fisher vectors for image categorization,” in *Proc. International Conference on Computer Vision*, pp. 1487-1494, 2011.
32. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proc. The Conference on Neural Information Processing Systems*, 2012.
33. S. Lazebnik, C. Schmid, and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2169-2178, 2006.
34. T. Leung and J. Malik, “Representing and Recognising the Visual Appearance of Materials using Three-dimensional Textons,” *International Journal of Computer Vision*, vol. 43, pp. 29-44, 2001.
35. F. Li, R. Fergus and P. Perona, “Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories,” in *Proc. 2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 178-178, 2004.
36. T. Lin, A. RoyChowdhury, and S. Maji, “Bilinear Convolutional Neural Networks for Fine-Grained Visual Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1309-1322, 2018.
37. L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M Pietikäinen, “From BoW to CNN: Two Decades of Texture Representation for Texture Classification,” *International Journal of Computer Vision*, vol. 127, no. 1, pp.74–109, 2019.
38. D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91-110, 2004.
39. B. S. Manjunath and W. Y. Ma, “Texture features for browsing and retrieval of image data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 837-842, 1996.
40. J. Mao and A.K. Jain, “Texture classification and segmentation using multiresolution simultaneous autoregressive models,” *Pattern Recognition*, vol. 25, no. 2, pp.173-188, 1992.
41. J. Y. Ng, F. Yang, and L. S. Davis, “Exploiting local features from deep networks for image retrieval,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015.
42. T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution Grey-Scale and Rotation Invariant Texture Classification with Local Binary Patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 971-987, 2002.
43. A.S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN Features off-the-shelf: an Astounding Baseline for Recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 23-28, 2014.
44. K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Visual Recognition,” in *Proc. International Conference on Learning Representations*, 2015.
45. J. Sivic, and A. Zisserman, “Video Google: a text retrieval approach to object matching in videos,” in *Proc. International Conference on Computer Vision*, pp. 1470-1477, 2003.
46. Y. Song, F. Zhang, Q. Li, H. Huang, L. J. O’Donnell and W. Cai, “Locally-Transferred Fisher Vectors for Texture Classification,” in *Proc. 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4922-4930, 2017.
47. M. Unser, “Sum and Difference Histograms for Texture Classification,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 1, pp. 118-125, 1986.
48. M. Varma and A. Zisserman, “A Statistical Approach to Material Classification Using Image Patch Exemplars,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, pp. 2032-2047, 2009.
49. F. M. Vilnrotter, R. Nevatia, and K. E. Price, “Structural Analysis of Natural Textures,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp.76-89, 1986.
50. J.S. Weszka, C.R. Dyer, and A. Rosenfeld, “A Comparative Study of Texture Measures for Terrain Classification,” *IEEE Trans. Systems, Man, Cybernetics*, vol. 6, pp. 269-285, 1976.
51. J. Xue, H. Zhang and K. Dana, “Deep Texture Manifold for Ground Terrain Recognition,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 558-567, 2018.
52. A. B. Yandex and V. Lempitsky, “Aggregating Local Deep Features for Image Retrieval,” in *Proc*. *International Conference on Computer Vision*, 2015.
53. Y. Yang and S. Newsam, “Spatial pyramid co-occurrence for image classification,” in *Proc. IEEE International Conference on Computer Vision*, 2011.
54. C. Yu, X. Zhao, Q. Zheng, P. Zhang, and X. You, “Hierarchical Bilinear Pooling for Fine-Grained Visual Recognition,” in: *Proc. European Conference on Computer Vision*, pp. 595-610, 2018.
55. B. Zhang, Y. Gao, S. Zhao, and J. Liu, “Local Derivative Pattern Versus Local Binary Pattern: Face Recognition With High-Order Local Pattern Descriptor,” *IEEE Trans. Image Process*., vol. 19, no. 2, pp. 533-544, 2010.
56. H. Zhang, J. Xue and K. Dana, “Deep TEN: Texture Encoding Network,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2896-2905, 2017.

1. X. Dong is supported by the Young Taishan Scholars Program. J. Dong is supported by National Natural Science Foundation of China (NSFC) (No. 60702014, 61271405 and 61401413) and the Ph.D. Program Foundation of Ministry of Education of China (No. 20120132110018). H. Zhou was supported by UK EPSRC under Grant EP/N011074/1, Royal Society-Newton Advanced Fellowship under Grant NA160342, and European Union’s Horizon 2020 research and innovation program under the Marie-Sklodowska-Curie grant agreement No 720325. (Corresponding authors: Xinghui Dong and Junyu Dong).

   X. Dong is with the Department of Computer Science, Ocean University of China, Qingdao, China (e-mail:dongxinghui@gmail.com).

   H. Zhou is with the School of Informatics, the University of Leicester, LE1 7RH, UK (e-mail: hz143@leicester.ac.uk).

   J. Dong is with the Department of Computer Science, Ocean University of China, Qingdao, China (e-mail: dongjunyu@ouc.edu.cn). [↑](#footnote-ref-0)