Texture Classification Using Sparse Representations by Learned Compound Dictionaries

J. Herredsvela, K. Engan, T. O. Gulsrud, and K. Skretting

University of Stavanger, Stavanger, Norway

jostein.herredsvela@uis.no

Abstract: A novel method for classification of texture images through sparse representation of image blocks is presented. The method enables us to classify multi-texture images by using one single learned compound dictionary. Promising results are presented for two two-texture images and one five-texture image. Comparisons with other known classification methods show that the classification performance of our method is good. The processing time for texture classification using our method is seen to be shorter than using the FTCM method, in which one dictionary is learned for each class.

1 Introduction

This paper treats the topic of texture classification of images by means of a new dictionary-based method.

The term *texture* is very difficult to define. In short it may be regarded as "what constitutes a macroscopic region" [1] in an image. It is important to note that "the notion of texture is undefined at the single pixel level but always associated with some set of pixels..." [2]. In our work no classical texture features are extracted from the images, e.g. local mean value, entropy, GLCM features, etc.

2 Materials and Methods

2.1 Dictionaries and Learning

Any N-dimensional vector can be written as a linear combination of $K \geq N$ vectors that span the space. A good *representation* of an N-dimensional signal vector $\mathbf{x}$ can often be obtained by linearly combining only a few of these K vectors. Mathematically, $\mathbf{x} = \mathbf{F}\mathbf{w} + \mathbf{n}$, where $\mathbf{F}$ is a learned frame/dictionary [3] in the form of an $N \times K$ ($K \geq N$) matrix, and where $\mathbf{w}$ is a sparse (i.e. most of the entries are zero) coefficient vector. $\mathbf{n}$ is the representation error vector. The columns of $\mathbf{F}$ are *dictionary vectors*. We emphasize that $\mathbf{F}$ must be *learned*, contrary to the case of orthogonal expansions, since a properly trained dictionary results in a small representation error even using a very sparse representation, if the test vectors are reasonably similar to the vectors used when learning the dictionary. Even choosing $K < N$ often yields good results. In general the representation error is larger for vectors that differ much from the vectors used when learning the dictionary.

For images, an image block consisting of a pixel and its neighborhood pixels can be reshaped into a (column) vector $\mathbf{x}$. Given a collection of L such vectors that are to be approximated using $\mathbf{F}$, an $N \times L$ data matrix $\mathbf{X}$ can be formed, of which the columns are the collection of vectors, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_L]$. Then the representation can be written

$$\mathbf{X} = \mathbf{F}\mathbf{W} + \mathbf{N},$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_L]$ and $\mathbf{N}$ is the representation error matrix.

$\mathbf{F}$ must be *learned* to represent $\mathbf{X}$ given the constraint that only S dictionary vectors are to be used in the representation of each column vector in $\mathbf{X}$. Our aim is to minimize the representation error $||\mathbf{X}-\mathbf{F}\mathbf{W}||$ subject to the coefficient matrix $\mathbf{W}$ being sparse [3].

The learning is performed using the Method of Optimal Directions (MOD) [3]. Starting from the initial dictionary, an iterate of $\mathbf{W}$ is found using a fast implementation of the Order Recursive Matching Pursuit (ORMP) algorithm [4]. Then a new dictionary $\mathbf{F}$ is calculated as

$$\mathbf{F} = \mathbf{X}\mathbf{W}^{\dagger}, \tag{1}$$

where $\mathbf{W}^{\dagger}$ is the pseudoinverse of $\mathbf{W}$. The procedure is repeated until there is little improvement in the representation error. The algorithm does not guarantee convergence to the optimal $\mathbf{F}$, but a good suboptimal solution is often reached.

2.2 Texture Classification and Compound Dictionaries

The FTCM (Frame Texture Classification Method) method of Skretting et al. [5, 6] performs texture classification by learning one dictionary for each texture class and then testing all the dictionaries on each pixel in the test image to see which one gives the best representation of the image block associated with the pixel, for a given sparsity. If the block is

best represented by the dictionary learned using, say, class no. i, its center pixel is assigned class i. The results presented are very good compared with other known methods, and therefore give good reasons to further explore the use of dictionaries/frames in texture classification. Our method uses only *one* compound dictionary in the learning and classification, as will be described next.

The following is performed for every texture class $i = 1, \ldots, c$: Assume that all the column vectors in an $N \times L_i$ training matrix $\mathbf{X}_i$ are extracted from a training image that represents only one texture class i. Each training vector is now labelled with a *label vector* $\mathbf{z}_i$ corresponding to this texture class. A labelled training matrix is formed as:

$$\mathbf{X}_i^{'} = \begin{bmatrix} \mathbf{X}_i \\ \cdots \\ \mathbf{Z}_i \end{bmatrix},$$

where $\mathbf{Z}_i$ is a single-class label matrix with all L_i columns equal to $\mathbf{z}_i$. An initial class-specific dictionary $\mathbf{F}_i$ is formed by randomly selecting K_i of the labelled data vectors.

Having c training matrices and c class-specific dictionaries, a total labelled $(N + c) \times L$ data matrix is created as follows:

$$\mathbf{X}^{'} = \begin{bmatrix} \mathbf{X}_1^{'}, \mathbf{X}_2^{'}, ..., \mathbf{X}_c^{'} \end{bmatrix} = \begin{bmatrix} \mathbf{X} \\ \cdots \\ \mathbf{Z} \end{bmatrix}.$$

Here $\mathbf{X}$ is the unlabelled data matrix and $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1, \mathbf{Z}_2, ..., \mathbf{Z}_c \end{bmatrix}$. An initial $(N + c) \times K$ *compound dictionary* $\mathbf{F}$ is formed as

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_c \end{bmatrix} = \begin{bmatrix} \mathbf{F}_x \\ \cdots \\ \mathbf{F}_z \end{bmatrix}.$$

Here $\mathbf{F}_x$ is the part of the dictionary corresponding to the unlabelled training data matrix $\mathbf{X}$ and $\mathbf{F}_z$ is the part corresponding to the label matrix $\mathbf{Z}$. $\mathbf{X}^{'}$ is a multi-class *training matrix*. We now have:

$$\mathbf{X}^{'} = \mathbf{F}\mathbf{W}' + \mathbf{N}.$$

$\mathbf{F}$ is now learned using the MOD algorithm. The fast ORMP algorithm used for vector selection assumes an input dictionary with normalized columns. We prefer to keep the original norm of each column of $\mathbf{F}$ by modifying, for every iteration, the coefficient matrix found by the ORMP algorithm.

The learned dictionary $\mathbf{F}$ can now be used for representing blocks from any of the textures used during training. This is exploited in the classification step.

We want to perform a texture classification of an $m \times n$ grayscale image consisting of c or fewer textures, using the learned dictionary $\mathbf{F}$. The texture label of each column is unknown and must be estimated.

Overlapping blocks from the test image are reshaped into column vectors in an unlabelled test data matrix $\mathbf{Y}$. Using $\mathbf{F}$, $\mathbf{Y}$ and its unknown label matrix $\mathbf{Z}_y$ can be written as

$$\begin{bmatrix} \mathbf{Y} \\ \cdots \\ \mathbf{Z}_y \end{bmatrix} = \begin{bmatrix} \mathbf{F}_x \\ \cdots \\ \mathbf{F}_z \end{bmatrix} \cdot \mathbf{W}_y^{'} + \mathbf{N}.$$

We see that $\mathbf{Y}$ can be represented using the unlabelled part $\mathbf{F}_x$ of $\mathbf{F}$ as

$$\tilde{\mathbf{Y}} = \mathbf{F}_x \mathbf{W}_y^{'} = \mathbf{F}_x^{'} \mathbf{W}_y = \mathbf{F}_x \mathbf{\Omega} \mathbf{W}_y,$$

where $\mathbf{F}_x^{'}$ is the normalized version of $\mathbf{F}_x$, used by the ORMP algorithm. $\mathbf{\Omega}$ is a diagonal $K \times K$ matrix with elements

$$\Omega_{jj} = \frac{1}{||\mathbf{f}_x^{(j)}||},$$

i.e. the inverse norms of the columns of $\mathbf{F}_x$.

$\mathbf{W}_y$ is found using the ORMP algorithm, and $\mathbf{W}_y^{'}$ is readily computed as

$$\mathbf{W}_y^{'} = \mathbf{\Omega} \mathbf{W}_y.$$

An estimate of $\mathbf{Z}_y$ can be written as

$$\tilde{\mathbf{Z}}_y = \mathbf{F}_z \mathbf{W}_y^{'}.$$

$\tilde{\mathbf{Z}}_y$ should now ideally contain all the correct labels corresponding to the columns in $\tilde{\mathbf{Y}}$, which also means of the corresponding (center) pixels in the original image. In practice, however, a direct pixel-wise classification will give a noisy result. The situation can be improved upon as follows: We reshape each row in $\tilde{\mathbf{Z}}_y$ into an $m \times n$ image. Image no. l is the reshaped l'th row of $\tilde{\mathbf{Z}}_y$. The c images are then smoothed using gaussian filters. Such smoothing is common in texture classification, and can be justified since texture is not a property which is defined pixel-wise. The resulting images are reshaped back into a new label matrix $\tilde{\mathbf{Z}}_y^s$.

The classification of each column vector $\tilde{\mathbf{z}}_y^s$ of $\tilde{\mathbf{Z}}_y^s$ is based on a simple distance measure:

$$class(\tilde{\mathbf{z}}_y^s) = \min_l ||\tilde{\mathbf{z}}_y^s - \mathbf{z}_l||,$$

where $\mathbf{z}_l$ is the label vector of texture class l.

Finally the resulting class vector is reshaped into an image of the same size as the original image. Any region of constant pixel value l in this image is a region of estimated texture class l.

3 Experiments and Results

The presented method has been tested on two two-texture images of size 256×512 pixels and one five-class image of size 256×256 pixels. The learning was performed on one separate 256×256-pixel image for each of the classes. Several block sizes were used. All textures were taken from the Brodatz album [7].

The set of label vectors $\mathbf{z}_i$, $i = 1, ..., c$, was selected as c-element orthogonal vectors of norm γ. The label

vectors are such that only element no. i is nonzero, i.e.

$$\mathbf{z}_i = [\underbrace{0, 0, \ldots}_{i-1 \text{ zeros}}, \gamma, \underbrace{0, \ldots, 0}_{c-i \text{ zeros}}]^T.$$

γ's total influence on the learning and classification is not yet fully analyzed. The idea is that it controls how much the selection of a vector of the wrong class is to be penalized during learning. The optimum value may vary with e.g. the coarseness or similarity of the textures to be learned and classified.

The first of the two-class images is presented in Figure 1(a). The misclassified areas obtained using different filter bandwidths are shown in Figures 1(b)-1(d). The image block size was 7×7 pixels, $S = 3$ vectors were used in the linear combination of frame vectors, and $K = 2 \cdot K_i = 50$, i.e. each initial class-specific dictionary was undercomplete. 100 iterations were used in the dictionary learning. The classification error performance for this image is given in Table 1. Two different block sizes were used: 7×7 and 9×9.

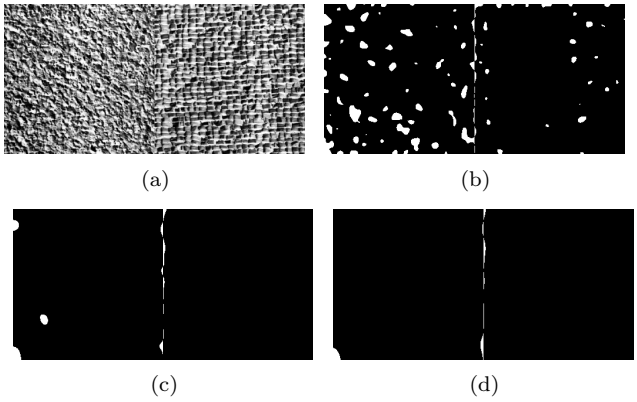

Figure 1: The first two-texture image and classification error results obtained using different smoothing filter bandwidths σ. (a): First texture image. (b): Result with $\sigma = 4$. (c): Result with $\sigma = 8$. (d): Result with $\sigma = 12$. The image block size was 7×7 pixels, $S = 3$, $\gamma = 100$, and $K = 2 \cdot K_i = 50$. 100 iterations were used.

Observe that the classification becomes better with increasing σ. Also observe that the classification performance at the boundary between the textures becomes worse with increasing σ. Classification errors in terms of percentage of misclassified pixels for this image are given in Table 1.

Table 1: Classification error performance in terms of percentage of wrongly classified pixels for the first texture image.

Block size	K_i	$\sigma = 4$	$\sigma = 8$	$\sigma = 12$
7×7	25	5.0%	0.8%	0.5%
9×9	41	6.5%	1.4%	0.7%

We see that the performance is better using the smaller block size. This is probably due to the texture being quite fine.

The second two-texture image consists of one relatively fine, irregular texture and one relatively coarse, irregular texture. The image is shown in Figure 2 together with obtained classification results. The image block size was 9×9 pixels, $S = 3$ vectors were used in the linear combination of frame vectors, and $K = 2 \cdot K_i = 114$. 200 iterations were used in the dictionary learning. The classification error performance for this image is shown in Table 2.

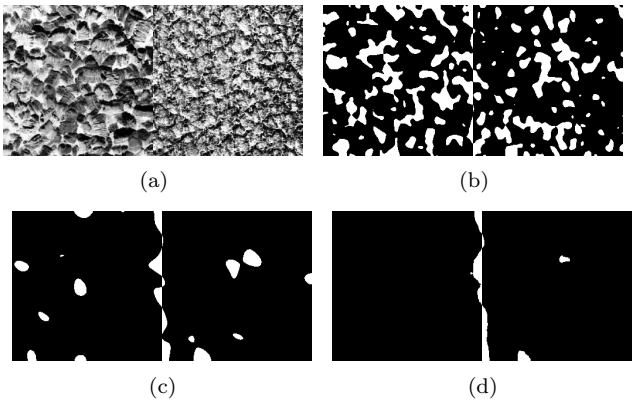

Figure 2: The second two-texture image and classification error results obtained using different smoothing filter bandwidths σ. (a): Second texture image. (b): Result with $\sigma = 4$. (c): Result with $\sigma = 12$. (c): Result with $\sigma = 20$. The image block size was 9×9 pixels, $S = 3$, $\gamma = 100$, and $K = 2 \cdot K_i = 114$. 200 iterations were used.

Classification errors in terms of percentage of misclassified pixels for this image are given in Table 2. Various parameter settings were used. We see that

Table 2: Classification error performance for the second two-texture image.

Block size	K_i	γ	$\sigma = 4$	$\sigma = 12$	$\sigma = 20$
5×5	25	100	31.8%	13.2%	8.4%
5×5	50	100	29.1%	10.8%	7.1%
5×5	100	100	28.5%	20.2%	20.6%
7×7	25	100	34.0%	29.8%	–
7×7	49	100	28.1%	14.0%	11.0%
7×7	25	500	28.6%	8.9%	4.5%
9×9	57	100	23.5%	4.5%	2.0%

increasing K does not necessarily improve the classification performance.

The FTCM method is able to classify this image very well using 5×5 image blocks: With $\sigma = 16$ an error rate of 1.2% is obtained. As can be seen from Table 2, we have to use 9×9 blocks to get comparable results with our method. Our best results are better than all the results listed in [8].

The five-texture image on which we have tested our method is shown in Figure 3 with obtained classification results. The parameters used were 7×7 block size, $S = 3$, $\gamma = 100$ and $K = 5 \cdot K_i = 200$. 100 iterations were used.

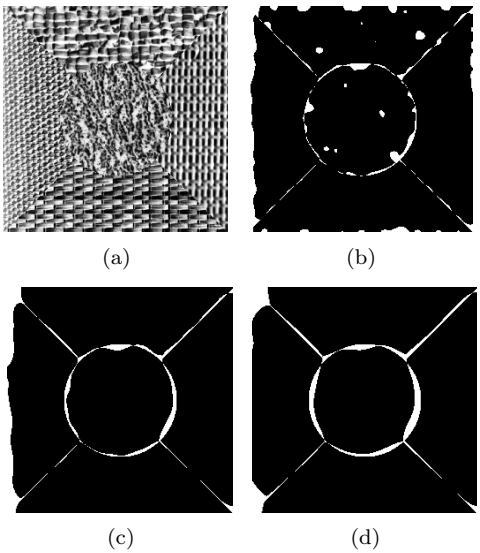

Figure 3: The five-texture image and classification error results obtained using different smoothing filter bandwidths σ. (a): Five-texture image. (b): Result with $\sigma = 4$. (c): Result with $\sigma = 8$. (d): Result with $\sigma = 12$. The parameters used were 7×7 block size, $S = 3$, $\gamma = 100$ and $K = 5 \cdot K_i = 200$. 100 iterations were used.

A summary of the classification performance obtained using 7×7 image blocks and $\gamma = 100$ with various parameter settings is given in Table 3.

Table 3: Classification error performance in terms of percentage of wrongly classified pixels for image Nat5c.

K_i	$\sigma = 4$	$\sigma = 8$	$\sigma = 12$
15	7.4%	5.7%	5.3%
25	7.0%	4.9%	5.2%
40	6.6%	5.2%	5.0%

We see that the classification result is quite good even using $K_i = 15$ and $\sigma = 4$. Also for this image, our results are better than those listed in [8].

For regular, fine textures, as in Figures 1(a) and 3(a) the proposed method works very well using relatively small dictionaries and little smoothing. For the coarser, more irregular textures in Figure 2(a) larger image blocks and dictionaries as well as more smoothing is necessary.

Finally we make a short direct comparison of our method and FTCM for image Nat5c. We used 7×7 image blocks and $S = 3$. In the FTCM case we used five class-specific dictionaries with $K = 25$. Using our method we used one labelled dictionary with, initially, $K_i = 25$, such that $K = 5 \cdot K_i = 125$. The learning was terminated after 100 iterations in both cases. The learning took quite much longer time using our method, but the classification took almost half the time FTCM did. The classification error results are shown in Table 4. We see that for this image the two methods yield comparable results when some smoothing is used.

Table 4: Direct comparison of classification error results for Nat5c obtained using FTCM and our proposed method.

Method	$\sigma = 4$	$\sigma = 8$	$\sigma = 12$
FTCM	4.3%	4.2%	5.7%
Prop. method	7.0%	4.9%	5.2%

4 Discussion and Conclusion

We have presented a new method for texture classification using sparse representation with one learned compound dictionary per classification task. Apart from the method being theoretically interesting, classification potentially becomes faster using the proposed method than the FTCM method with similar parameters. The performance is overall somewhat worse than the FTCM, but the difference is image-dependent. The difference seems to be largest for coarse, irregular textures.

An obvious drawback of the method is that the dictionaries become very large when many classes are to be classified. This results in very long learning times and memory problems, especially for coarse, irregular textures for which large K_i and N may be needed. The method is therefore best suited for few classes.

The authors would like to thank Professor Kenneth Kreutz-Delgado from the University of California, San Diego for valuable discussions and ideas.

References

[1] H. Tamura, S. Mori, and Y. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8:237–247, 1978.

[2] M. Unser and M. Eden. Nonlinear operators for improving texture segmentation based on features extracted by spatial filtering. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(4):804–815, July/August 1990.

[3] K. Engan, S. O. Aase, and J. H. Husøy. Multi-frame compression: Theory and design. *Signal Processing*, (80):2121–2140, October 2000.

[4] M. Gharavi-Alkhansari and T. S. Huang. A fast orthogonal matching pursuit algorithm. In *Int. Conf. on Acoust. Speech and Signal Proc.*, pages 1389–1392, Seattle, U.S.A, May 1998.

[5] K. Skretting. *Sparse signal representation using overlapping frames.* PhD thesis, Norges Teknisk-naturvitenskapelige universitet (NTNU)/Høgskolen i Stavanger, October 2002. Available at *http://www.ux.his.no/ karlsk/*.

[6] K. Skretting and J. H. Husøy. Texture classification using sparse frame based representations. *Eurasip JASP*. Accepted for publication.

[7] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. New York: Dover, 1966.

[8] M. Randen and J. H. Husy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, April 1999.