Texture Classification using Naïve Bayes Classifier

Ayman M Mansour

Department of Communication, Electronics and Computer Engineering,
Tafila Technical University, Tafila, Jordan

Summary
This paper presents a texture classification algorithm using Independent Component Analysis and Naïve Bayes Classifier. Naïve Bayes is one of the most effective and efficient classification algorithms. Naïve Bayes classifiers still tend to perform very well under unrealistic assumption. Especially for small sample sizes, naive Bayes classifiers can outperform the more powerful classifiers. Texture features are extracted using Independent Component Analysis and then classified by Naïve Bayes Classifier. Experiments were performed in order to evaluate the performance of the proposed classifier. It consists of texture images from the Describable Textures Dataset (DTD) and Brodatz album. Experimental results show that the proposed algorithm has an encouraging performance. The Naïve Bayes Classifier produces a very accurate classification results.

Key words:
Naïve Bayes, feature, Independent Component, Classifier.

1. Introduction

Texture classification involves deciding what texture category an observed texture image belongs to. In order to accomplish this, a priori knowledge of the classes to be recognized is needed. Once this knowledge is available and the texture features are extracted, classical pattern classification techniques can be used in order to do the classification task.

The texture classification problem is normally divided into the two sub problems of feature extraction and classification [1-4]. Many recent methods have been developed to extract textural features, which includes statistical, model-based, and signal processing methods [1,5-8].

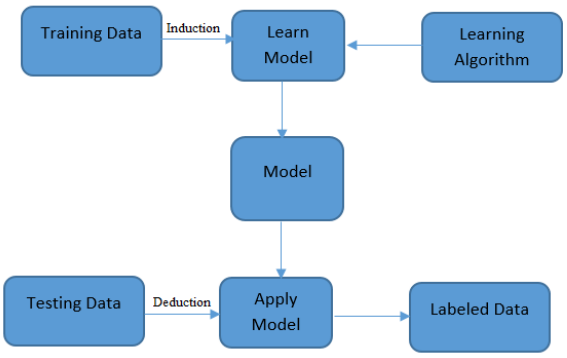
In this paper we introduce Independent Component Analysis (ICA) of textured images as a componential technique for creating appropriate feature vectors. Then, the classification task is performed used Naïve Bayes classifier. Naive Bayes Classifier has been successfully used in many recent application [9-17]. It is one of most popular statistical learning systems.

Independent component analysis (ICA) is a relatively new signal processing and data analysis technique. Independent Component Analysis is used widely in literature for blind source separation and has been applied in many application e.g., in wireless communications, biomedical signal processing texture classification and data mining [18-27].

The ICA based approach is different from existing methods in that it produces a new data dependent bank which called basis functions in order to be used in texture classification. These basis functions are able to capture the inherent properties of textured images. Feature vectors are obtained from training data using ICA algorithm. In order to perform this, it is required to provider training data containing the appropriate structure. In this paper, the independent components basis functions are computed using a fast ICA algorithm by Hyvainen [28]. The algorithm was derived based on neg-entropy. This fast ICA algorithm was empirically shown to be 10 to 100 times faster than many ICA algorithms.

2. Methodology

Texture classification process involves four phases: Data gathering phase, Data preprocessing phase, the learning phase and the classification phase. In data gathering, the training and test set will be obtained from a texture images databases. The second phase is to pre-process the images, including sampling, whitening centering and dimensionality reduction. In the learning phase, the target is to build a model. The last step is using the remaining of the pre-processed data to test the model. A test set is used to determine the accuracy of the model. Usually, the given texture image set is divided into training and test sets, with training set used to build the model and test set used to validate it (Fig.1).



Manuscript received January 5, 2018
Manuscript revised January 20, 2018

Fig. 1 The learning process: training and testing.

2.1 Training stage

The texture image can be represented by a weighted linear combinations of basis function as shown in Fig. 2.

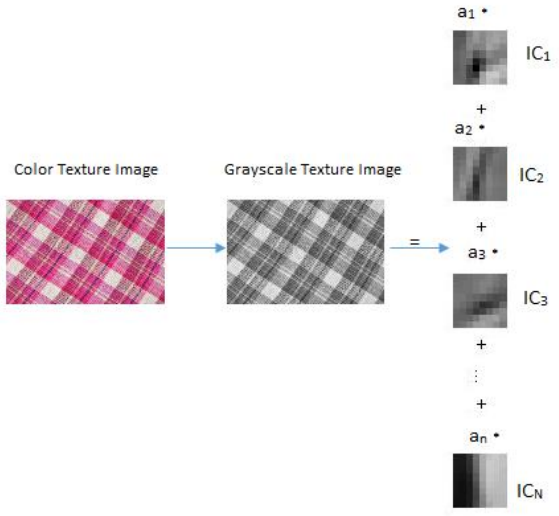


Fig. 2 An image can be expressed as a linear weighted sum of basis images. The weights are the independent components.

The basis functions ICi are statistically independent of each other. The texture image of size (n x n) is illustrated in equation (1)

$$\mathbf{x} = \sum_{i=1}^{N} IC_i \mathbf{a}_i, \qquad \textbf{(1)}$$

Where $N = n^2$, IC_i is a the i^{th} basis function vector and a_i is independent component of basis function i

Rewriting equation (1) in matrix form leads to equation (2),

$$x = IC\ A, \qquad \textbf{(2)}$$

where $IC = [\mathbf{IC_1, IC_2, \ldots, IC_N}]$ and $A = [a_1, a_2, \ldots, a_n]^T$. This is called the ICA model.

The goal of ICA is to use the images x_t, t = 1,..., M, to estimate a matrix $W = IC^{-1}$ such that the linear transform makes the components a_i, i = 1,..., N, of A as statistically independent as possible .

Equation (3) presents the form the FastICA algorithm used in this paper as ICA algorithm

$$\mathbf{w}^{+} = E(\mathbf{x}g(\mathbf{w}^T\mathbf{x})) - E(g'(\mathbf{w}^T\mathbf{x}))\mathbf{w}$$

$$\mathbf{w}^{*} = \frac{\mathbf{w}^{+}}{\| \mathbf{w}^{+} \|} \qquad \textbf{(3)}$$

Where $g(w^T x)=(w^T x)^3$ and $g'(w^T x)=3(w^T x)^2$ are based on kurtosis algorithms [28]. FastICA will start by selecting random values of the weight matrix W and each iteration update the weight based on equation (3) until convergence.

The texture image is initially transferred to a column vector by screening each raw in the image and concatenate it in the column vector. Due to this process part of the two-dimensional correlations is destroyed. This can be solved in the training stage by rotating the texture images by random angle θ, where θ is between 0° and 180°. Each rotated image will be used in the training process.

Using k training texture images, (m x m) sub images will be generated. The size of the window should be large enough to contain sensible visual information and at the same time small enough to introduce generality in the data. This tradeoff can be solved selecting the appropriate window size and shape that maximize the classification accuracy.

The used window sizes in this paper is (8×8). The used shape is the rectangular one, where a sample window of size (m x m) with top corner positioned at (x_0,y_0) is obtained by multiplying the original image function x with window function.

$$w(x, y, x_0, y_0) = \begin{cases} 1 \ \textit{if} \ x_0 \le x \le x_0 + m - 1 \ \textit{and} \ y_0 \le y \le y_0 + m - 1 \\ 0 \ \textit{otherwise} \end{cases} \qquad \textbf{(1)}$$

After multiplication, the area outside the window is discarded, leaving (m x m) image window. To ensure that no texture dominates the basic functions, each training texture image is normalized and made zero mean before sampling. In addition, every generated sub-image subtracted its individual local mean value. If this is not done, one of the resulting basis functions will represent the mean intensity value of the training data, which is of little value in the bank context. Fig. 3 summarizes the steps used to generate the training vectors set from training texture images.

✓ **Normalize training texture images and make zero mean.**

✓ **Sampling loop:** ***for i = 1 to number of training images,***

** Rotation loop: for j = 1 to number-of-directions (in our case 4)*

- Rotate training texture number *i* a random angle *θj*
- Select the given number of *(m x m)* sized patches form random locations
- Subtract from each patch its mean value
- Represent samples as columns, and store in a matrix vectors.

✓ **End**

Fig. 3 Steps to generate training texture vectors

The resulted training vectors will be input to ICA algorithm in order to produce basis functions bank. The proposed algorithm used to generate feature vectors is shown in Fig.4.

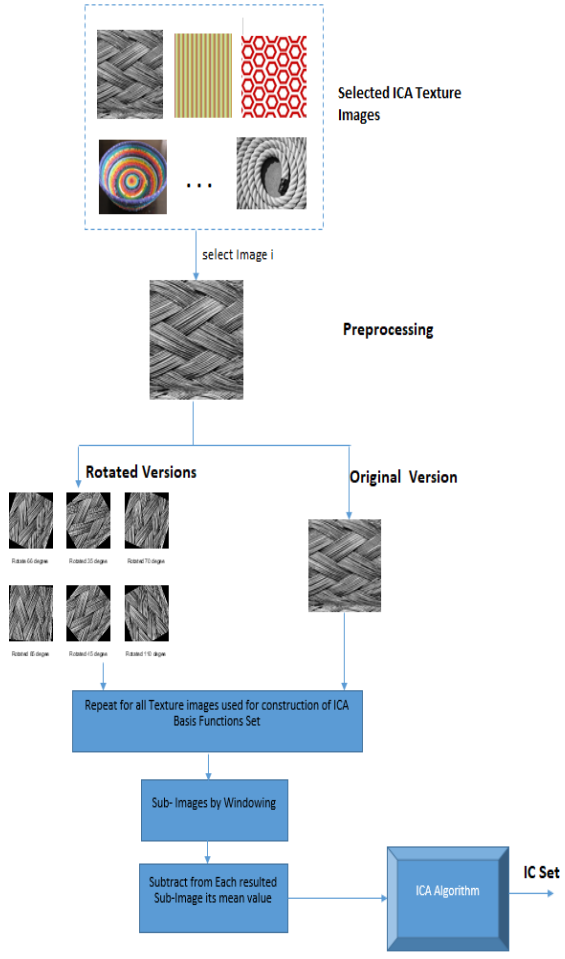


Fig. 4 The algorithm for generating IC basis functions.

The generated basis function bank is shown in Fig.5.

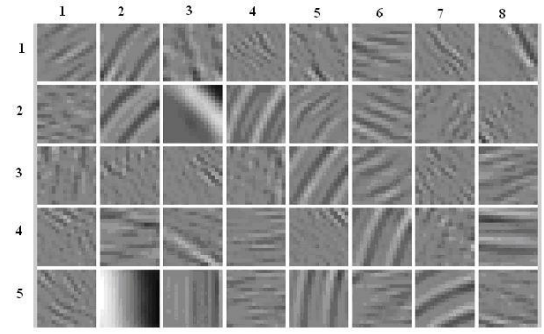


Fig.5 Example of (8 x 8) ICA basis functions for texture.

The basis function bank will be used to generate feature vectors and the assigned class label as shown in Fig.6.

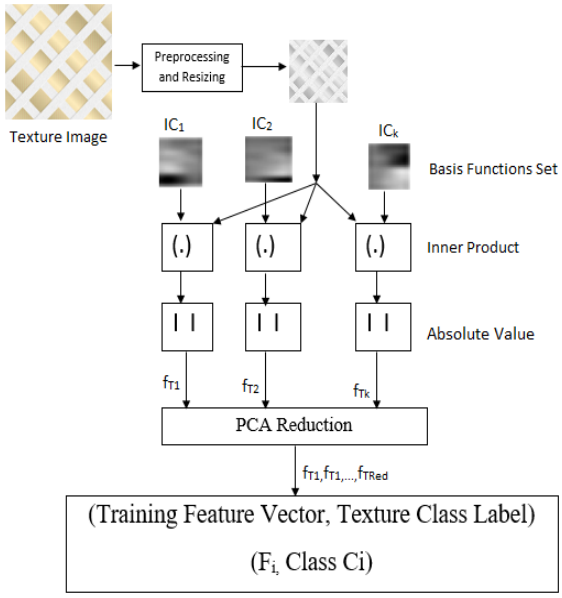


Fig. 6 Feature extraction using ICA.

The input vector x comes directly from the gray-level values of the (m ×m) window resulting from the input image. The preprocessing consists of two operations; Centering and Whitening. Centering the data to the origin is done by subtracting the mean of the data from each x,

$x := x - E\{x\}$.

This is done to simplify the estimation of the correlation matrix of x. Whitening, is a transformation of the data so that the components of the data are uncorrelated and have unit variance, i.e,

$$\mathbf{z} = \mathbf{V}\mathbf{x}, \text{ so that } E\{zz^T\} = \mathbf{I}, \qquad (2)$$

where **I** is the identity matrix, **V** is the whitening matrix

and **z** is the whitened data.

The inner product between the vector x and each of the independent Components (IC's) banks, each with specific frequency and orientation characteristics will be computed. If the bank consists of k independent components, the result from the inner product is k values called f_{Ti} where i=1,…,k .

$$f_{Ti} = |x^T . ICi| \quad (3)$$

The size of the k ICs will be the same size as the input image patch (mx m). Each v corresponds to specific texture properties. Thus, each input image will have high value in regions corresponding to textures which are tuned to the IC, and low value corresponding to textures which are not tuned to. If v was the selected feature vector, it has the size (mx1).

Principal Component Analysis (PCA) is a transformation that aims to represent N-dimensional data in an orthogonal uncorrelated L-dimensional space such that the mean square error (MSE) between the original data x and the representation s in the new PCA basis is minimized, That is,

$$\textbf{Minimize } E\{ ||x-s||2 \} \quad (4)$$

The principal component vectors, are given by the eigenvectors of the correlation matrix of x, Rx, such that

$$\mathbf{Rx} = E\{\mathbf{xx^T}\} = \mathbf{V\lambda V^T} \quad (5)$$

Where x is assumed to be zero mean. The columns of V contain the eigenvectors, and the diagonal matrix **λ** contains the corresponding eigenvalues.

The resulted dimension is truncated into a L < N. This reduces the space and time complexity and the degree of freedom. The reduction is done by rearranging the eigenvectors and eigenvalues and dropping the eigenvectors corresponding to small eigenvalues.

2.2 Testing stage

Texture classification is basically the problem of classifying pixels in a texture image according to their textural cues. The proposed system is shown in Fig 7 .The test image will be divided to (8 by 8) sized windows. The constructed sub-images will be preprocessed and feature vector will be extracted. The feature vector will be classified using Naïve Bayes classifier to the appropriate class.

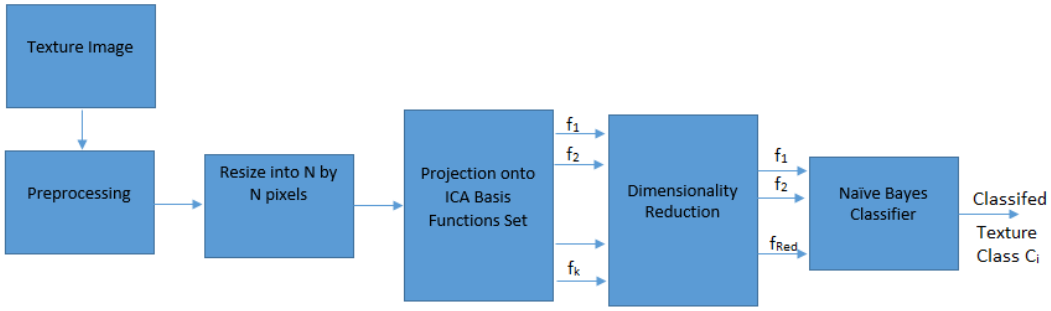


Fig.7 Proposed Texture classification approach using ICA and Naïve Bayes Classifier.

The list of the sets of feature values and its corresponding class are given to the classifier with the constituted training set. The model is trained using Naïve Bayes Classifier learning algorithm. When a new record values, previously unseen test record, is presented to the classifier, the class can be predicted based on the training instances.

The probability that a record with feather vector F= $\{F_1, F_2, \dots, F_n\}$ belongs to class Ci, Where $C_i \in \{C_1, C_1, \dots, C_k\}$

$$p(C_i|\boldsymbol{F}) = p(C_i|F_1, F_2, \dots, F_n) = \frac{p(F_1, F_2, \dots, F_n|C_i).P(C_i)}{p(F_1, F_2, \dots, F_n)} \quad (6)$$

$p(C_i|\boldsymbol{F})$ is the posterior probability of class (C_i) given features ($F_1, F_2, \dots, F_n$).

$P(C_i)$is the prior probability of class (C_i).

$p(F_1, F_2, \dots, F_n|C_i)$ is the likelihood which is the probability of features ($F_1, F_2, \dots, F_n$) given class (C_i).

P(x) is the prior probability of features ($F_1, F_2, \dots, F_n$).

Calculating $p(C_i|\boldsymbol{F})$ is the main aim in Naïve Bayes Classifier. Specifically, we want to find the value of C_i that maximize $p(C_i|F_1, F_2, \dots, F_n)$. Since $p(F_1, F_2, \dots, F_n)$ is constant number for all k values, then equation (9) can simply say

$$p(C_i|\boldsymbol{F}) = p(C_i|F_1, F_2, \dots, F_n) \propto (p(F_1, F_2, \dots, F_n|C_i).P(C_i)) \quad \textbf{(7)}$$

Where

$$P(C_i) = \frac{N_{C_i}}{N_T} \quad \textbf{(8)}$$

N_{ci} is count of samples from class Ci.
N_T is Count of all samples.

Naive Bayes classifier assumes conditional independence that the effect of the value of a feature (F) on a given class (C) is independent of the values of other predictors [4]. Then the $p(F_1, F_2, \dots, F_n|C_i)$ can be written as

$$p(F_1, F_2, \dots, F_n|C_i) = \prod_{j=1}^{n} p(F_j|C_i) \quad \textbf{(9)}$$

For numerical features, the Gaussian Naive Bayes algorithm assumes distribution of features to be Gaussian or normal, i.e., the probability density function for the Gaussian distribution is defined by two parameters (mean and standard deviation).

$$p(F_j|C_i) = \frac{1}{\sqrt{2\pi\sigma_{ji}^2}} e^{-\frac{(F_j-\mu_{ji})^2}{2\sigma_{ji}^2}} \quad \textbf{(10)}$$

Where,

$$\mu_{ji} = Sample\ Mean = \frac{1}{K}\sum_{K=1}^{K} F_K \quad \textbf{(11)}$$

$$\sigma_{ji} = Sample\ Variance = \sqrt{\frac{1}{K-1}\sum_{K=1}^{K}(F_K - \mu_{ji})^2} \quad \textbf{(12)}$$

Finally the Naïve Bayes calculate the posterior probability for each class. Choose value of Ci that maximizes P(Ci | F1, F2, …, Fn) is equivalent to choosing value of Ci that maximizes P(F1, F2, …, Fn|Ci) P(Ci).
The class with the highest posterior probability is the predicted one. The estimated class $\widehat{C}_i$ Corresponding to F is

$$The\ Predicted\ Class\ \widehat{C}_i = \max_i\ p(C_i|F_1, F_2, \dots, F_n), 1 \le i \le K \quad \textbf{(13)}$$

3. Experiments

In order to evaluate the previous methods, textures subset (28 texture image) were taken from the Describable Textures Dataset (DTD) [29][30] as shown in Fig.8. DTD is a texture database consists of 5640 images. These images are classified to 47 categories inspired from human perception. There are 120 images for each category. Image sizes range between 300x300 and 640x640. These texture images provide us with different basis functions properties (features).Changing any one of the 28 images will produce new basis functions. The resulted basis functions are shown in Fig.5. The construct the ICA basis functions is achieved after applying extensive dimension reduction as provided by PCA

Fig.8 Training textures from the Describable Textures Dataset (DTD).

The texture classifier was trained on randomly selected portions of (8×8) sub images of texture images that are not included in the test images. The total number of texture images used in the experimental part is 193 images.

The performance measurements used for this paper were recall, precision, classifier F1 rating and accuracy. Recall (R) is the ratio of the relevant data among the retrieved. Precision (P) is the ratio of the accurate data among the retrieved data. Their formulas are given as follow:

$$Recall(R) = \frac{T_P}{T_P+F_N}$$ if TP+FN > 0, otherwise undefined. **(14)**

$$Precision(P) = \frac{T_P}{T_P+F_p}$$ if TP+Fp > 0, otherwise undefined. **(15)**

Classifier F1 rating is the harmonic mean of the classifier recall and the precision. It is given as

$$F_1 = \frac{2*P*R}{P+R}$$ **(16)**

where R represents the recall, , and P represents the precision

Accuracy, which indicates the fraction of correctly classified samples among all the samples,
obtained by:

$$Accuracy = \frac{T_P+T_N}{T_P+T_N+F_P+F_N}$$ **(17)**

Where,

- True Positive (TP) represents the number of positive data that is correctly labeled by a classifier
- True Negative (TN) represents the number of negative data that is correctly labeled by a classifier
- False Positive (FP) represents the number of positive data that is incorrectly labeled by a classifier
- False Negative (FN) represents the number of negative data that is incorrectly labeled by a classifier

Macro-averaged measure has been used in this multi label classification. The macro-averaged results can be computed as indicated by:

Macro precision:

$$Precision_{macro} = \frac{1}{n}\sum_{i=1}^{n} Precision(Ci)$$ **(18)**

Macro Recall:

$$Recall_{macro} = \frac{1}{n}\sum_{i=1}^{n} Recall(Ci)$$ **(19)**

Macro F-measure:

$$F_{macro} = \frac{1}{n}\sum_{i=1}^{n} F(Ci)$$ **(20)**

Where n is the number of the classes Ci

10-fold cross validation has been used in this paper. In 10-fold the training set will be randomly splitted into 10's that have approximately the same size. Then the classifier will be trained using (8) subsets. One of the two remaining subsets will be used for validation and the last for testing.

This process will be repeated 10 times, while a different subset is used for testing and validation. Table 1 shows the average accuracy for 2-Fold, 4-Fold, 6-Fold, 8-Fold and the 10-Fold cross-validation

Table 1: Accuracy for 2-Fold, 4-Fold, 6-Fold, 8-Fold and the 10-Fold cross-validation.

K-Fold	Accuracy (%)
2-Fold	91.5044 %
4-Fold	93.2743 %
6-Fold	93.2743 %
8-Fold	92.3894 %
10-Fold	99.4819 %

The experiments were performed with input window sized (8×8). Using this size, the best classification accuracy has been achieved as shown in Table 2. When the texture boundary is more as in texture image shown in Fig.9, a larger window size was considered to be disadvantageous. The correct classification rate (%) of rectangular shaped window and circular shaped window of radius R is shown in Fig. 9.

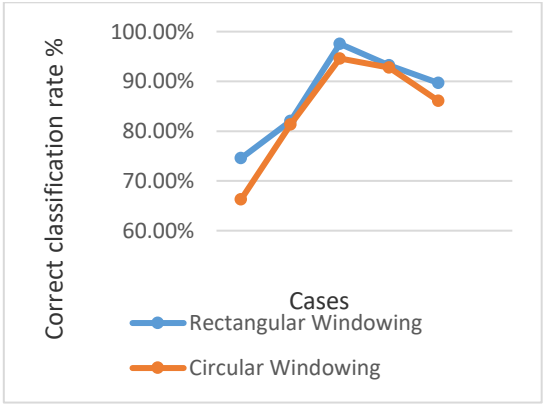


Fig.9 Correct classification rate of rectangular window and circular window.

Table 2: Correct classification rate for different window size.

Window size	Correct classification rate (%)
3x3	74.56%

5x5	82.06%
8x8	97.54%
12x12	93.24%
17x17	89.69%

As the number of features in that matrix was 40 features, we used PCA dimensionality reduction method and we selected around 23% of the features (9 features) with the highest importance.

There are many different reduction techniques available including Principal Component Analysis (PCA), and Chi-Squared for feature selection and reduction. We used the Describable Textures Dataset to compare PCA and Chi-Squared on a reduced number of dimensions varying between 1 and 9 dimensions. Fig. 10 depicts the F1 rating of these two methods. Clearly, PCA F_1 measurement of PCA outperformed the other method.

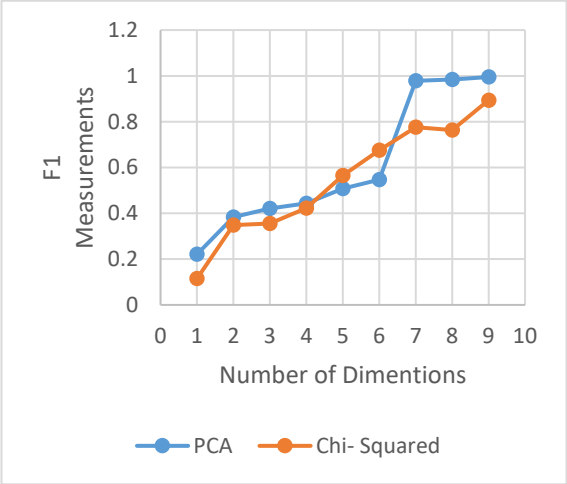


Fig.10 F1 Measurement for PCA and Chi-Squared.

As stated before, 5640 Texture from the database were used to train and test the developed system. Table 3 shows the experimental result of 193 texture images. The 193 texture images are from 4 different classes. The proposed algorithm has a recognition accuracy of 99.4%.

Table 3: Overall performance results.

Total Number of Instances	193	
Correctly Classified Instances	192	99.4819 %
Incorrectly Classified Instances	1	0.5181 %
Kappa statistic	0.9924	
Mean absolute error	0.0467	
Root mean squared error	0.0856	
Relative absolute error	17.0269 %	
Root relative squared error	23.1491 %	

Another performance indicated by confusion matrix is shown in Table 4. This confusion matrix was built based on data testing. We constructed the confusion matrix for each texture class (Texture 1, Texture 2, Texture 3, Texture 4). The confusion matrix has the form shown in Table 4,

Table 4: Confusion matrix.

	Texture 1	Texture 2	Texture 3	Texture 4
Texture 1	69	1	0	0
Texture 2	0	76	0	0
Texture3	0	0	30	0
Texture 4	0	0	0	17

The performance measurements result is shown in Table 5.

Table 5: Performance Measurements Result.

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.986	0	1	0.986	0.993	Texture 1
1	0.009	0.987	1	0.993	Texture 2
1	0	1	1	1	Texture 2
1	0	1	1	1	Texture 2

We have also applied the proposed algorithm to several images of composite textures of different texture patterns with size 256×256 pixels and 256 grey levels. Composite textures images from the Brodatz album [31], [32] are used in this part of the experiment in order to compare the achieved result with the performance of other classifier proposed in the literature. Fig. 11 and Fig. 12 show examples of composite texture images.

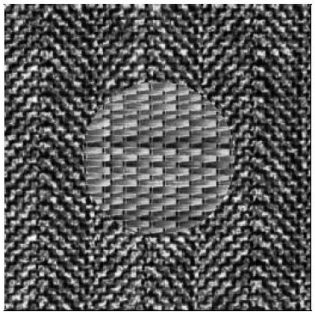


Fig. 11 Two-texture images D55D17 used in experiments (Size 256 X 256).

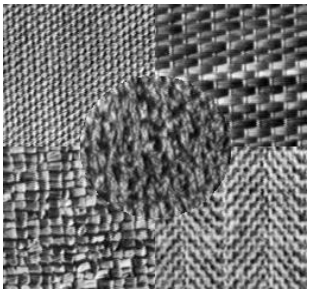


Fig.12 Five-texture imagesD55D17D84D77D24 used in experiments.

All of the images give an error percentage of below 5% which is a good result. Notice that the more texture boundaries there are, the more difficult decisions must be made, resulting in an increasing number of misclassified pixels. No boundary pixels seem to be well distinguished by the proposed algorithm.
The results obtained from the proposed Naïve Bayes Classifier was compared by the results achieved using ICA, SVM, and using the filtering method (Gaber filter and wavelet). Some of these is shown in Table 6 [33],[34] and [35] .The proposed methodology a large classification improvement was observed.

Table 6: Accuracy rates (%) using different classifiers and different texture image.

Classifier / images	ICA	SVM	Wavelet	Proposed method
D55D17	99.038	----	88.88	99.47
D55D17D84D77D24	95.71	80.7	----	96.25

4 Conclusions

We have demonstrated that the Naïve Bayes Classifier with ICA are able to capture the inherent properties of textured images and correctly labeled it. The excellent classification error rate achieved in the experiments confirm that the using of Naïve Bayes Classifiers well-suited for texture classification. So the proposed algorithm can be characterized as reliable.

References

[1] P. Cavalin and L. S. Oliveira, "A Review of Texture Classification Methods and Databases," 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images,2017

[2] Jasjit S. Suri, Majid Mirmehdi, Xianghua Xie, Handbook of Texture Analysis, World Scientific, 2008.

[3] A Olson, D., Data Mining Models, Business Expert Press 2016.

[4] Ian H Witten; Eibe Frank; Mark A Hall; Christopher J Pal, Data Mining : Practical Machine Learning Tools and Techniques, Fourth Edition, Elsevier Science and Technology Books, Inc., 2017

[5]

[6] S. Wang and G. Wang, "Texture classification by multifractal spectrum and barycentric coordinates of bit planes of wavelet coefficients," in IET Image Processing, vol. 11, no. 12, pp. 1205-1209, 12 2017.

[7] T. H. Rassem, A. A. Alsewari and N. M. Makbol, "Texture image classification using wavelet completed local binary pattern descriptor (WCLBP)," 2017 Seventh International Conference on Innovative Computing Technology (INTECH), Luton, 2017, pp. 15-20.

[8] C. Di Ruberto, "Histogram of Radon transform and texton matrix for texture analysis and classification," in IET Image Processing, vol. 11, no. 9, pp. 760-766, 9 2017.

[9] K. Meshkini and H. Ghassemian, "Texture classification using Shearlet transform and GLCM," 2017 Iranian Conference on Electrical Engineering (ICEE), Tehran, 2017, pp. 1845-1850.

[10] R. Bayindir, M. Yesilbudak, M. Colak and N. Genc, "A Novel Application of Naive Bayes Classifier in Photovoltaic Energy Prediction," 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 2017, pp. 523-527.

[11] K. Nirmala, N. Venkateswaran and C. V. Kumar, "HoG based Naive Bayes classifier for glaucoma detection," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, 2017, pp. 2331-2336.

[12] M. Granik and V. Mesyura, "Fake news detection using naive Bayes classifier," 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), Kiev, 2017, pp. 900-903.

[13] W. B. Zulfikar, M. Irfan, C. N. Alam and M. Indra, "The comparation of text mining with Naive Bayes classifier, nearest neighbor, and decision tree to detect Indonesian swear words on Twitter," 2017 5th International Conference on Cyber and IT Service Management (CITSM), Denpasar, 2017, pp. 1-5.

[14] Y. P. Chen, C. H. Liu, K. Y. Chou and S. Y. Wang, "Real-time and low-memory multi-face detection system design based on naive Bayes classifier using FPGA," 2016 International Automatic Control Conference (CACS), Taichung, 2016, pp. 7-12.

[15] N. Sharma and M. Singh, "Modifying Naive Bayes classifier for multinomial text classification," 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, 2016, pp. 1-7.

[16] Y. Suresh, "Software quality assessment for open source software using logistic & Naive Bayes classifier," 2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, 2016, pp. 267-272.

[17] R. M. Moraes and L. S. Machado, "A Fuzzy Binomial Naive Bayes classifier for epidemiological data," 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Vancouver, BC, 2016, pp. 745-750.

[18] S. C. Hsu, I. C. Chen and C. L. Huang, "Image classification using pairwise local observations based Naive Bayes classifier," 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Hong Kong, 2015, pp. 444-452.

[19] H. Martin, W. Izquierdo, M. Cabrerizo and M. Adjouadi, "Real-time R-spike detection in the cardiac waveform through independent component analysis," 2017 IEEE Signal Processing in Medicine and Biology Symposium (SPMB), Philadelphia, PA, USA, 2017, pp. 1-7.

[20] A. K. Maddirala and R. A. Shaik, "Separation of Sources From Single-Channel EEG Signals Using Independent Component Analysis," in IEEE Transactions on Instrumentation and Measurement, vol. 67, no. 2, pp. 382-393, Feb. 2018.

[21] L. Feng and R. Sun, "Dynamic kernel independent component analysis approach for fault detection and diagnosis," 2017 Chinese Automation Congress (CAC), Jinan, China, 2017, pp. 2193-2197.

[22] D. Wei, X. Li, C. Lei and W. Wang, "Process monitoring using independent component analysis based on gradient

and newton iteration methods," 2017 Chinese Automation Congress (CAC), Jinan, China, 2017, pp. 3182-3187.
[23] S. Çinar, E. C. Mengüç and N. Acir, "Automatic removal of ocular artefacts in EEG signal by using independent component analysis and Artificial Neural Network," 2017 Medical Technologies National Congress (TIPTEKNO), Trabzon, Turkey, 2017, pp. 1-4.
[24] Z. Gu, J. Huangfu and L. Ran, "Blind separation of human motions based on independent component analysis," 2017 International Applied Computational Electromagnetics Society Symposium (ACES), Suzhou, 2017, pp. 1-2.
[25] P. Jaraut, G. C. Tripathi, M. Rawat and P. Roblin, "Independent component analysis for multi-carrier transmission for 4G/5G power amplifiers," 2017 89th ARFTG Microwave Measurement Conference (ARFTG), Honololu, HI, 2017, pp. 1-4.
[26] P. Addabbo, C. Clemente and S. L. Ullo, "Fourier independent component analysis of radar micro-Doppler features," 2017 IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace), Padua, 2017, pp. 45-49.
[27] A. K. Maddirala and R. A. Shaik, "Separation of Sources From Single-Channel EEG Signals Using Independent Component Analysis," in IEEE Transactions on Instrumentation and Measurement, vol. 67, no. 2, pp. 382-393, Feb. 2018.
[28] W. L. Hwang, K. S. Lu and J. Ho, "Constrained Null Space Component Analysis for Semiblind Source Separation Problem," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 2, pp. 377-391, Feb. 2018.
[29] Aapo Hyvainen, Erkki Oja, Juha Karhunen. “Independent Component Analysis”, (1st ed.). New York: John wiley &sons, 2001.
[30] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed and A. Vedaldi, "Describing Textures in the Wild," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 3606-3613.
[31] https://www.robots.ox.ac.uk/~vgg/data/dtd/
[32] http://www.ux.uis.no/~tranden/brodatz.html
[33] P.Brodatz, Textures: A Photographic Album for Artists and Designers. New York: Dover, 1966.
[34] C. S. Lu, P.C. Chung, and C.F. Chen. “Unsupervised Texture Segmentation via Wavelet Transform”. Pattern Recognition, Vol 30(5),PP 729-742, 1997.
[35] Se Hyun Park, Kwang In Kim, Keechul Jung , and Hang Joon Kim. “Support Vector Machines for Texture Classification”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24 (11), pp1542-1550 ,2002.
[36] D. A. Al Nadi and A. M. Mansour, "Independent Component Analysis (ICA) for texture classification," 2008 5th International Multi-Conference on Systems, Signals and Devices, Amman, 2008, pp. 1-5.

Dr. Ayman M Mansour received his Ph.D. degree in Electrical Engineering from Wayne State University in 2012. Dr. Mansour received his M.Sc degree in Electrical Engineering from University of Jordan, Jordan, in 2006 and his B.Sc degree in Electrical and Electronics Engineering from University of Sharjah, UAE, in 2004. He graduated top of his class in both Bachelor and Master. Currently, Dr. Mansour is an assistant professor in the Department of Communication, Electronics and Computer Engineering, Tafila Technical University, Jordan. He is also the director of energy research center in Tafila Technical University. His areas of research include communication systems, multi-agent systems, fuzzy systems, data mining and intelligent systems. He conducted several researches in his area of interest. Dr. Mansour is a member of IEEE, Michigan Society of Professional Engineers, IEEE Honor Society (HKN), Society of Automotive Engineers (SAE),