Contents lists available at ScienceDirect

Journal of Food Engineering

journal homepage: www.elsevier.com/locate/jfoodeng

Fruit classification using computer vision and feedforward neural network

Yudong Zhang [a,*], Shuihua Wang [a], Genlin Ji [a], Preetha Phillips [b]

[a] School of Computer Science and Technology, Nanjing Normal University, Nanjing, Jiangsu 210023, China
[b] School of Natural Sciences and Mathematics, Shepherd University, Shepherdstown, WV 25443, USA

ARTICLE INFO

Article history:
Received 18 October 2013
Received in revised form 30 June 2014
Accepted 1 July 2014
Available online 9 July 2014

Keywords:
Color histogram
Unser's texture analysis
Shape feature
Feedforward neural network
Fitness-scaled chaotic artificial bee colony

ABSTRACT

Fruit classification is a difficult challenge due to the numerous types of fruits. In order to recognize fruits more accurately, we proposed a hybrid classification method based on fitness-scaled chaotic artificial bee colony (FSCABC) algorithm and feedforward neural network (FNN). First, fruits images were acquired by a digital camera, and then the background of each image were removed by split-and-merge algorithm. We used a square window to capture the fruits, and download the square images to 256 $\times$ 256. Second, the color histogram, texture and shape features of each fruit image were extracted to compose a feature space. Third, principal component analysis was used to reduce the dimensions of the feature space. Finally, the reduced features were sent to the FNN, the weights/biases of which were trained by the FSCABC algorithm. We also used a stratified *K*-fold cross validation technique to enhance the generation ability of FNN. The experimental results of the 1653 color fruit images from the 18 categories demonstrated that the FSCABC–FNN achieved a classification accuracy of 89.1%. The classification accuracy was higher than Genetic Algorithm–FNN (GA–FNN) with 84.8%, Particle Swarm Optimization–FNN (PSO–FNN) with 87.9%, ABC–FNN with 85.4%, and kernel support vector machine with 88.2%. Therefore, the FSCABC–FNN was seen to be effective in classifying fruits.

1. Introduction

Fruit classification is a difficult and important task in supermarkets, since it is necessary for the cashier to know the categories of a particular fruit in order to determine its price. The use of barcodes has mostly resolved this problem for packaged products; however, most consumers want to pick their products by themselves. Some fruits cannot be packaged using barcodes, and thus must be weighted. One solution can be to issue codes for every fruit, but the tedious memorization of barcodes can lead to errors in pricing. Another solution can be to issue the cashier an inventory with pictures and codes; however, flipping over the inventory booklet can be overall time consuming (Rocha et al., 2010).

Scholars have proposed several effective solutions based on computer vision and machine learning to address similar problem during the last decade. Baltazar et al. (2008) first applied data fusion to nondestructive image of fresh intact tomatoes, followed by a three-class Bayesian classifier. Pennington and Fisher (2009) used clustering algorithm for fruits and vegetables classification. Pholpho et al. (2011) used visible spectroscopy for classification of non-bruised and bruised longan fruits, and combined the principal component analysis (PCA), Partial Least Square Discriminant Analysis and Soft Independent Modeling of Class Analogy to develop classification models. VeggieVision (Bolle et al., 1996) was a supermarket produce recognition system that consisted of an integrated scale and image system with a user-friendly interface. Hong et al. (2006) employed morphological examination to separate the walnut and hazelnut into three groups. Zhang and Wu (2012) chose the Max-Wins-Voting SVM with Gaussian RBF kernel to recognize the different categories of fruits. Cano Marchal et al. (2013) established an expert system based on computer vision to estimate the content of impurities in olive oil samples. Fan et al. (2013) used a two hidden layers of back-propagation artificial neural network to predict the texture characteristics from extrusion food surface images. Omid et al. (2013) developed an intelligent system based on combined fuzzy logic and machine vision techniques for grading of egg using parameters as defects and size of eggs.

The aforementioned techniques may have one or several of the following four shortcomings. First, they need extra sensors such as a gas sensor, an invisible light sensor, or a weight sensor. Second, the classifier is not suitable to all fruits, viz., it can only recognize

* Corresponding author.
E-mail addresses: zhangyudong@njnu.edu.cn (Y. Zhang), wangshuihua@njnu.edu.cn (S. Wang), glji@njnu.edu.cn (G. Ji), preetha.phillips@gmail.com (P. Phillips).

http://dx.doi.org/10.1016/j.jfoodeng.2014.07.001
0260-8774/© 2014 Elsevier Ltd. All rights reserved.

the varieties of the same category. Third, the recognition systems are not robust because different fruit images may have similar or identical color and shape features. Fourth, customers complain that there is a misclassification among the fruits (Seng and Mirisaee, 2009).

The objective of this paper is to propose a novel fruit classification system based on computer vision, with aim of solving above four shortcomings to the utmost degree. First, we use merely a digital camera, getting rid of other complicated sensors. Second, the proposed classifier is expected to recognize as many types of fruits as possible. In this study, 18 types of fruits are included. Third, we capture not only conventional color and shape features, but also the influential texture features. Finally, the proposed classifier is expected to have good accuracy by the feedforward neural network (FNN), since it is a powerful tool among supervised classifiers and it can also classify nonlinear separable patterns and approximate an arbitrary continuous function (Coulibaly and Evora, 2007).

However, finding the optimal weights/biases of FNN is a difficult task because the traditional gradient-based optimization algorithms, such as back-propagation (BP) algorithms, are easily trapped in local extrema. Recently, there has been many global optimization algorithms available to train the FNN, such as Genetic Algorithm (GA), Simulating Annealing (SA) algorithm, and Particle Swarm Optimization (PSO). Unfortunately, the BP, GA, SA, and PSO algorithms all demand expensive computational costs, and can be easily trapped into the local best. This means that the result would probably end up without finding the optimal weights/biases of the FNN. In this study, we use the artificial bee colony (ABC) algorithm to find the optimal weights/biases of FNN. ABC algorithm was originally presented by Karaboga et al. (2004) under the inspiration of collective behavior on honey bees with better performance in function optimization problems compared with GA, differential evolution, and PSO (Karaboga and Basturk, 2008). As it is known, standard global optimization techniques conduct only one search operation in each iteration. For example, the PSO carries out a global search at the beginning stage and local search in the ending stage (Zhang and Wu, 2011); nevertheless, the ABC features the advantage in that it conducts both a global search and local search in each iteration. As a result, the probability of finding the optimal is significantly increased, which effectively avoids local optima to a large extent. In order to improve the performance of ABC, Zhang et al. embedded the fitness scaling strategy and chaotic theory into the ABC, naming their new method as fitness scaled chaotic artificial bee colony (FSCABC) (Zhang et al., 2011b).

The structure of the following paper is organized as: Section 2 contains the procedure of data processing, including the preprocessing, feature extraction and reduction, and stratified cross validation techniques. Section 3 describes the classifier FNN, introduces in the FSCABC algorithm to train the weights/biases of FNN, and summarizes the steps of our fruit recognition system. Section 4 presents the experimental results step by step. We also compared the results of our method FSCABC–FNN with GA–FNN, PSO–FNN, ABC–FNN, and kernel support vector machine (kSVM). Section 5 discussed the results. Final Section 6 is devoted to conclusions and future research.

2. Data processing

The acronyms used in the paper are listed in Table 1.

Table 1
Glossary.

Acronyms	Full expressions
ABC	Artificial bee colony
BP	Back-propagation
DE	Differential evolution
FNN	Feedforward neural network
FSCABC	Fitness-scaled chaotic artificial bee colony
GA	Genetic algorithm
kSVM	kernel support vector machine
LB	Lower bound
MBP	Momentum BP
MSE	Median square error
PCA	Principal component analysis
PSO	Particle swarm optimization
RBF	Radial basis function
SA	Simulating annealing
SVM	Support vector machine
UB	Upper bound

2.1. Preprocessing

First, we used image segmentation techniques to remove the background since our research only concentrate on the fruits. We chose the split-and-merge algorithm (Damiand and Resch, 2003; Xiao et al., 2001), which was based on a quadtree partition of an image. The method started at the root of the tree that represents the whole image. If the four sub-squares were found inhomogeneous, then it was split into four son-squares (the splitting process), and so on so forth. Conversely, if four sub-squares were homogeneous, they should be merged as several connected components (the merging process). The node in the tree was a segmented node. This process continued recursively until no further splits or merges were possible. Second, we used a square window to capture the fruit, making the fruit lying in the center of the window. Finally, we downsampled the square images to 256×256. Although the downsampling degraded the image quality, it made the algorithm performed faster. Fig. 1 shows all the steps of preprocessing.

2.2. Feature extraction and reduction

We proposed a composite feature space based on color, texture, and shape features of fruits. For a 256×256 image, we extracted 64 color features, 7 texture features, and 8 shape features. The feature extraction can be seen as the dimension reduction procedure. The original tri-color image vector space has a $256 \times 256 \times 3 = 196,608$ dimensions. By feature extraction operation, the size of the feature vector space is only $64 + 7 + 8 = 79$. Afterwards, PCA was employed to reduce the number of features further by the criterion that the reduced features should cover at least 95% variance of original features. Fig. 2 illustrates the flowchart of feature extraction and reduction.

2.2.1. Color discretization and histogram

So far, the color histogram is employed to represent the distribution of colors in an image (Siang Tan and Mat Isa, 2011). It represents the number of pixels that have colors in a fixed list of color range that span the image's color space (Maitra and Chatterjee, 2008). For monochromatic (grayscale) images, the set of possible color values is sufficiently small that each of those colors may be placed on a single range; then the histogram is merely the count of pixels that have each possible grayscale value. For color images using RGB space, the histogram is produced first by discretization of the colors in the image into a number of bins, and counting the number of image pixels in each bin.

Fig. 3 shows the illustration of color discretization. For each channel, the intensity value before discretization varies from 0 to 255 (Fig. 3a), hence, there are totally $256 \times 256 \times 256 = 2^{24} = 16{,}777{,}216$ different colors. Then, we use 4 bins (Fig. 3b) to represent each color channel. Bin 0, 1, 2, 3 denotes intensities 0–63, 64–127, 128–191, and 192–255, respectively. This step is termed

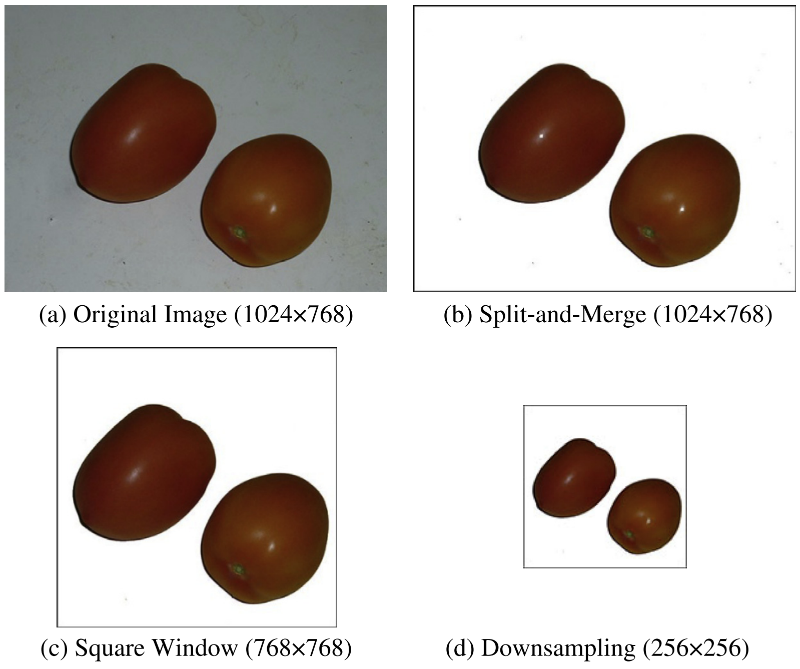

(a) Original Image (1024×768)
(b) Split-and-Merge (1024×768)
(c) Square Window (768×768)
(d) Downsampling (256×256)

Fig. 1. Three steps in preprocessing.

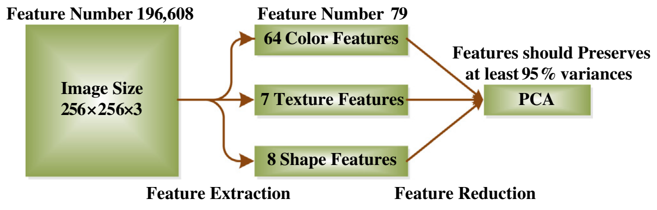


Fig. 2. Flowchart of feature extraction and reduction.

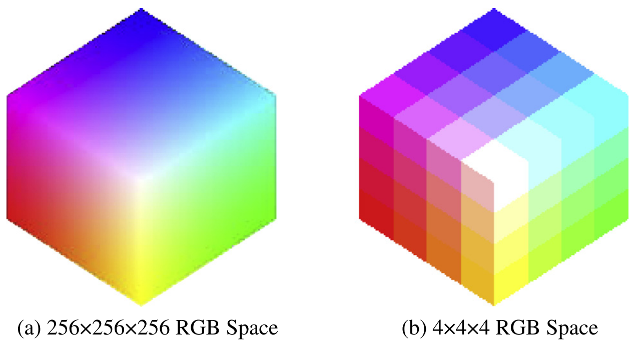


Fig. 3. Illustration of color discretization that reduces (a) 16,777,216 types of color to only and (b) 64 types of colors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

“color discretization”. After that, there are totally $4 \times 4 \times 4 = 64$ different types of colors.

Fig. 4(a) shows a “rainbow-rose” image with RGB channels. Fig. 4(b) is the color-discretized version of (a). Fig. 4(c) illustrates the histogram of (b). The x-axis denotes the index of the 64 colors, and y-axis denotes the number of pixels.

The histogram provides a compact summarization of the distribution of colors in an image. It is relatively invariant with translation and rotation about the viewing axis. By comparing histograms signatures of two images and matching the color content of one image with the other, the color histogram is well suited for the problem of recognizing an object of unknown position and rotation within a scene, and it is employed as an important source of feature extraction in this study.

2.2.2. Unser's texture features

Gray level co-occurrence matrix (Ou et al., 2014; Raheja et al., 2013) and local binary pattern (Jia et al., 2014; Nosaka and Fukui, 2014) are good texture descriptors; however, they are excessively time consuming. In this study, we chose the Unser's texture feature vector. Unser proved that the sum and difference of two random variables with same variances are de-correlated and the principal axes of their associated joint probability function are defined. Therefore, we use the sum s and difference d histograms for texture description (Unser, 1995). The non-normalized sum and difference associated with a relative displacement (δ_1, δ_2) for an image I are defined as:

$$s(k,l;\delta_1,\delta_2) = I(k,l) + I(k+\delta_1,\ l+\delta_2) \quad (1)$$

$$d(k,l;\delta_1,\delta_2) = I(k,l) - I(k+\delta_1,\ l+\delta_2) \quad (2)$$

The sum and difference histograms over the domain D are defined as:

$$h_s(i;\delta_1,\delta_2) = \mathrm{card}((k,l) \in D, s(k,l;\delta_1,\delta_2) = i) \quad (3)$$

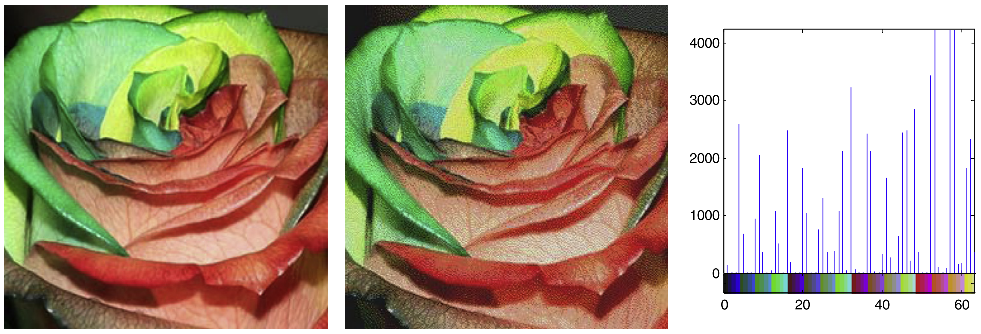
$$h_d(j;\delta_1,\delta_2) = \mathrm{card}((k,l) \in D, d(k,l;\delta_1,\delta_2) = j) \quad (4)$$

Next, seven indexes can be defined based on the sum and difference histogram. Those indexes and their corresponding formulae are listed in Table 2.

2.2.3. Shape features

In this study, we proposed eight mathematical morphology based measures listed in Table 3. The measures can be extracted using following three steps. Table 3 listed shape measures and their corresponding meanings.

Step 1: Extract “area”, “perimeter”, and the “Euler number” features directly from the object;
Step 2: Create a convex hull using Graham Scan method (Lou et al., 2012) which is the smallest convex polygon that covers



(a) Rainbow-Rose Image; (b) Reduced to 4*4*4=64 colors; (c) Color Histogram

Fig. 4. Illustration of color histogram of rainbow-rose image.

Table 2
Seven Unser's texture measures.

Measure	Symbol	Formula
Mean	μ	$\mu = (1/2) \times \sum_i i h_s(i; \delta_1, \delta_2)$
Contrast	C_n	$C_n = \sum_j j^2 h_d(j; \delta_1, \delta_2)$
Homogeneity	H_g	$H_g = \sum_j (1/(1+j^2))\, h_d(j; \delta_1, \delta_2)$
Energy	E_n	$E_n = \sum_i h_s(i; \delta_1, \delta_2)^2 \sum_j h_d(j; \delta_1, \delta_2)^2$
Variance	σ^2	$\sigma^2 = (1/2) \times (\sum_i (i - 2\mu)^2 h_s(i; \delta_1, \delta_2) + \sum_j j^2 h_d(j; \delta_1, \delta_2))$
Correlation	C_r	$C_r = (1/2) \times (\sum_i (i - 2\mu)^2 h_s(i; \delta_1, \delta_2) - \sum_j j^2 h_d(j; \delta_1, \delta_2))$
Entropy	H_n	$H_n = -\sum_i h_s(i; \delta_1, \delta_2) \log(h_s(i; \delta_1, \delta_2)) - \sum_j h_d(j; \delta_1, \delta_2) \log(h_d(j; \delta_1, \delta_2))$

Table 3
Eight morphology based shape measures.

Measure	Symbol	Meaning
Area	A_r	The actual number of pixels inside the object
Perimeter	P_r	The distance around the boundary of the object
Euler	E_l	The Euler number of the object
Convex	C_n	The number of pixels of the convex hull
Solidity	S_l	The proportion of area to convex hull
Minor Length	M_n	The length of the minor axis of the ellipse
Major length	M_j	The length of the major axis of the ellipse
Eccentricity	E_c	The eccentricity of the ellipse

the object, then extract the "convex area" and "solidity" features;
Step 3: Create an ellipse that has the same second-moments as the object, then extract the "minor length", "major length", and "eccentricity" features.

2.2.4. Feature reduction

In total, there are 79 features (64 color features + 7 texture features + 8 shape features) extracted from a prescribed image. Those excessive features increase computation time and storage memory, which sometimes causes the classification process to become more complicated and even decrease the performance of the classifier. A strategy is necessary to reduce the number of features used in classification.

PCA is an efficient tool to reduce the dimension of a data set consisting of a large number of interrelated variables while retaining the most significant variations (Kwak, 2008). It is achieved by transforming the data set to a new set of ordered variables according to their degree of variance or importance. This technique has three effects: (1) it orthogonalizes the components of the input vectors so that they are uncorrelated with each other, (2) it orders the resulting orthogonal components so that those with the largest variation come first, and 3) it eliminates the components in the data set that contributes the least variation (Lipovetsky, 2009). It should be noted that the input vectors should be normalized to have zero mean and unity variance before performing PCA. The normalization is a standard procedure. Details about PCA appears in Ref. (Jackson, 1991). The readers can use the "PCA" command by the Matlab platform to perform a standard PCA operation.

2.3. Stratified K-fold cross validation

After feature reduction by PCA, we divide the data into training and test set. The training set is used for train the weights/biases of the classifier, meanwhile the test set is used as test the performance of classifier. In order to enhance the generation capability of the classifier, we used the cross validation technique on the training set.

Typically, a statistical model that deals with the inherent data variability is inferred from the training set, and employed by statistical learning machines for the automatic construction of classifiers. The model has a set of adjustable parameters that are estimated in the learning phase using a set of examples. Nevertheless, the learning machine must ensure a reliable estimation of the parameters and consequently good generalization, i.e. correct responses to unseen examples, including classifying new images correctly. Hence, the learning device must efficiently find a trade-off between its complexity, which is measured by several variables, such as the effective number of free parameters of the classifier and the feature input space dimension, and the information on the problem given by the training set (e.g. measured by the number of samples).

Cross validation methods are usually employed to assess the statistical relevance of the classifiers. It consists of four types: Random subsampling, *K*-fold cross validation, leave-one-out validation, and Monte Carlo Cross-Validation (Pereira et al., 2011). The *K*-fold cross validation is applied due to its simple and easy properties, while using all data for training and validation. The mechanism is to create a *K*-fold partition of the whole dataset, repeat *K* times to use $K - 1$ folds for training and a left fold for validation, and finally average the error rates of *K* experiments. The schematic diagram of 5-fold cross validation is shown in Fig. 5.

The *K* folds can be purely random partitioned; however, some folds may have quite different distributions from other folds. Therefore, the stratified *K*-fold cross validation was employed, in which every fold has nearly the same class distributions (May et al., 2010). The folds are selected so that the mean response value is approximately equal in all the folds. In the case of a dichotomous classification, this means that each fold contains roughly the same proportions of the two types of class labels. Another challenge was to determine the number of folds. If *K* is set too large, the bias of the true error rate estimator will be small, but the variance of

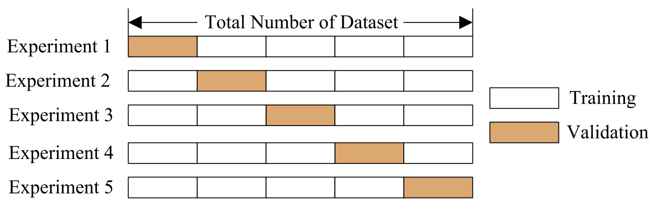


Fig. 5. A fivefold cross validation.

the estimator will be large and the computation will be time consuming. Alternatively, if K is set too small, the computation time will decrease, the variance of the estimator will be small, but the bias of the estimator will be large (Armand et al., 2007). In this study, we empirically determined K to be 5 through the trial-and-error method.

3. FSCABC–FNN

In last section, we extracted and reduced the features from the fruits pictures. Now we need to submit the feature data to the classifier. We will describe a novel classification technique dubbed as FSCABC–FNN.

3.1. FNN weights optimization

FNN is chosen as the classifier because it is widely used in pattern classification, and it does not need any information about the probability distribution and the a priori probabilities of different classes (Llave et al., 2012). The training vectors were presented to the FNN, which is then trained in batch mode. The general one-hidden-layer model of FNN devises a structure illustrated in Fig. 6.

It is clearly perceived that there are three layers contained in FNN (Amiri et al., 2008; Zhang et al., 2011c): input layer, hidden layer, and output layer. Nodes of each adjacent layer are connected completely and directly to form the links. Each link has a weighted value that presents the relational degree between two nodes (Goel and Pal, 2009). In the next paragraphs, we shall discuss how to choose the optimal weights of FNN, and convert it to an optimization problem.

3.1.1. Encoding strategy

Let N_I represents the number of input neurons, N_H the number of hidden neurons, and N_O the number of output neurons. Suppose ω_1 and ω_2 represents the connection weight matrix between the input layer and hidden layer, between the hidden layer and the output layer, respectively.

Fig. 7 shows the formation of the weight matrix ω_1 and ω_2. The encoding style can be presented as the combination of the vectorization of the (ω_1, ω_2).

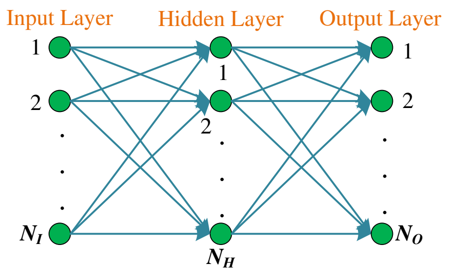


Fig. 6. Structural architecture of one-hidden-layer FNN.

$$\omega = [\mathcal{V}(\omega_1), \mathcal{V}(\omega_2)] = \left[\underbrace{\omega_1(1,1),\ldots\omega_1(N_I,N_H)}_{N_I * N_H}, \underbrace{\omega_2(1,1),\ldots\omega_2(N_H,N_O)}_{N_H * N_O}\right] \tag{5}$$

where $\mathcal{V}$ represents the vectorization operation.

3.1.2. Fitness function

Subsequently, we can infer the fitness function, i.e., the training process to update these weighted values, which can be divided into four steps (Sakthivel et al., 2010):

(1) The outputs of all neurons in the hidden layer are calculated by

$$y_j = f_H\left(\sum_{i=1}^{N_I}\omega_1(i,j)x_i\right) j = 1,2,\cdots,N_H \tag{6}$$

Here x_i denotes the ith input value, y_j denotes the jth output of the hidden layer, and f_H is referred to as the activation function of hidden layer (Crone and Kourentzes, 2010), usually a sigmoid function as follows:

$$f_H(x) = \frac{1}{1+\exp(-x)} \tag{7}$$

(2) The outputs of all neurons in the output layer are given as follows:

$$O_k = f_O\left(\sum_{j=1}^{N_H}\omega_2(j,k)y_j\right) k = 1,2,\ldots,N_O \tag{8}$$

Here f_O denotes the activation function of output layer, usually a line function. All weights are assigned with random values initially (Zanaganeh et al., 2009), and are modified by the delta rule according to the learning samples traditionally.

(3) The error is expressed as the median square error (MSE) of the difference between output and target value

$$E_l = \text{mse}\left(\sum_{k=1}^{N_O}(O_k - T_k)\right) l = 1,2,\ldots N_S \tag{9}$$

where T_k represents the kth value of the authentic values which are already known to users, and N_S represents the number of samples (Poursamad, 2009).

(4) The fitness function is written as the average MSE

$$f(\omega) = \sum_{l=1}^{N_S} E_l \tag{10}$$

The goal is to minimize this fitness function $f(\omega)$, viz., force the output values of each sample approximate to corresponding target values.

In a word, the weights of FNN are regarded as the variables, and average MSE between output and target is regarded as the fitness function. The weights optimization problem is to find the best weights that can minimize the average MSE.

3.2. FSCABC algorithm

To train the weights/biases of FNN, we introduced in the ABC algorithm. It has been proven to perform better than GA, DE and PSO (Karaboga and Basturk, 2008) with respect of several benchmark test functions. Based on ABC, Zhang et al. (2011a,b)

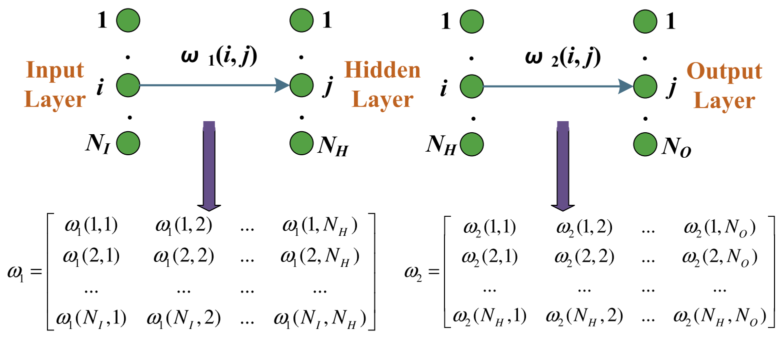


Fig. 7. The connection weight matrix between (a) input layer and hidden layer, (b) hidden layer and output layer.

had proposed the FSCABC that is proven to perform better than ABC. In what follows we will describe FSCABC in depth.

3.2.1. Power-rank fitness scaling

Fitness scaling converts the raw fitness scores that are returned by the fitness function to values in a range that is suitable for the selection function. The selection function uses the scaled fitness values to select the bees of the next generation. The selection function assigns a higher probability of selection to bees with higher scaled values.

There exist bundles of fitness scaling methods. The most common scaling techniques are linear scaling, rank scaling, power scaling, top scaling, etc. Among those fitness scaling methods, power scaling finds a solution nearly the most quickly due to improvement of diversity, but it suffers from instability (Korsunsky and Constantinescu, 2006). Meanwhile, rank scaling shows stability on different types of tests (Wang et al., 2010). Suppose

$$r_i = \text{rank}(f_i|[f_1, f_2, \dots f_N]) \tag{11}$$

where f_i is the original fitness function of ith individual bee, r_i its corresponding rank, and N the number of population. Then, the power-rank scaling method was described as:

$$\bar{f}_i = \frac{r_i^k}{\sum_{i=1}^{N} r_i^k} \tag{12}$$

where r^k represents r raised to the power of k, and $\bar{f}$ the scaled fitness value. The strategy contains a three-step process. First, all bees are sorted to obtain their corresponding ranks. Second, powers are computed for exponent k. Third, the scaled values are normalized by dividing the sum of the scaled values over the entire population.

3.2.2. Chaotic operator

The chaotic theory pointed out that minute changes in initial conditions steered subsequent simulations towards radically different final results, rendering long-term prediction impossible in general (Zhang et al., 2013). Sensitive dependence on initial conditions is not only observed in complex systems, but even in the simplest logistic equation. In the well-known logistic equation (Singh and Sinha, 2010):

$$c_{n+1} = 4 \times c_n \times (1 - c_n) \tag{13}$$

$$c_0 \in (0, 1) \ \&\ c_0 \notin \{0.25, 0.5, 0.75\} \tag{14}$$

where c_n represents chaotic number series. A very small difference in the initial value of c would give rise to a large difference in its long-time behavior as shown in Fig. 8(a and b). The track of chaotic variable c_n can travel ergodically over the whole space of interest. Fig. 8(c–e) indicates that the series c_n will lose chaotic property it its initial value is one of 0.25, 0.5, and 0.75.

The span of c_n falls within (0, 1). We define another chaotic number series φ_n that falls within the range (−1, 1) as:

$$\varphi_n = 2 * c_n - 1 \tag{15}$$

Random parameters in standard ABC are generated by pseudo-random number generators, which cannot ensure the ergodicity. Therefore, chaotic number series (13) and (15) are employed to replace the random parameters, with the aim to improve the performance of standard ABC. The detailed steps of FSCABC are listed below.

3.2.3. Procedures of FSCABC

Step 1: *Initialization:* Suppose N represents the number of populations, ω_{ij} represents the ith ($1 \leqslant i \leqslant N$) solution candidate at jth epoch. We initialize the population of solutions ω_{ij} with $j = 0$

$$\omega_{i0} = LB + c_i \times (UB - LB) \quad (i = 1, \dots, N) \tag{16}$$

here c_i represents the chaotic number series in (13), and LB and UB the lower and upper bounds, respectively. Then, evaluate the population via the prescribed fitness function f in (10)

$$f_{i0} = f(\omega_{i0}) \tag{17}$$

Step 2: Repeat and let $j = j + 1$;

(a) Produce new solutions (food source) υ_{ij} in the neighborhood of ω_{ij} for the employed bees using

$$\upsilon_{ij} = \omega_{ij} + \varphi_{ij}(\omega_{ij} - \omega_{neighbor(i),j}) \tag{18}$$

Here φ_{ij} is a chaotic random number in the range [−1, 1] calculated by (15). Evaluate the new solutions. Apply the greedy selection process between ω_{ij} and υ_{ij}

$$\{\omega_{ij}\} = \text{Select } N \text{ best candidates from} \{\omega_{ij}, \upsilon_{ij}\} \text{based on their fitness values} \tag{19}$$

(b) Produce new solutions (onlooker) υ_{ij} for the onlookers from the solutions ω_{ij} by (18), selected depending on the probability of $\bar{f}_{ij}$, and evaluate them. Apply the greedy selection process for the onlookers between ω_{ij} and υ_{ij} according to (19);

(c) Produce new solutions (scout). Determine the abandoned solution (the worst candidate) and replace it with a new randomly produced solution x_i for the scout using the equation

$$\omega_{worst,j} = \min_i(\omega_{ij}) + c_{ij} * \left(\max_i(\omega_{ij}) - \min_i(\omega_{ij})\right) \tag{20}$$

Here c_{ij} is a chaotic random number in [0, 1] generated by (13).

(d) Go to Step 2 until termination criteria met.

Step 3: Output the final result

$$\omega^* = \arg\max_{\omega}(f_{ij}) \tag{21}$$

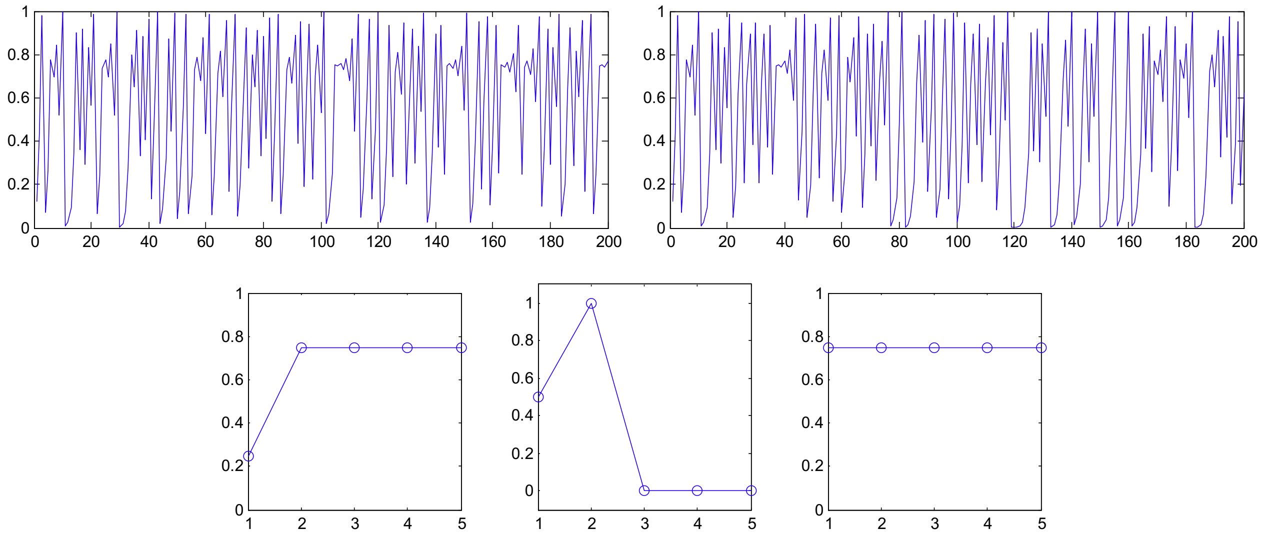


Fig. 8. Chaotic number series generated by logistic equation: (a) $c_0 = 0.12345678$; (b) $c_0 = 0.12345679$. Specific initial values will make the series lose chaotic property: (c) $c_0 = 0.25$; (d) $c_0 = 0.5$; (e) $c_0 = 0.75$.

3.3. Fruit recognition system

The flowchart of the proposed fruit recognition system is shown in Fig. 9. The pseudo codes of the recognition system are written as follows:

Step 1. The input is a database of 1653 color images consisting of 18 categories of fruits, and each image size is $1024 \times 768 \times 3$ (length = 1024, width = 768, and color channel = 3).
Step 2. *Preprocessing:* Remove background. Crop and resize each image to 256×256 size. Retain 3 color channels.
Step 3. *Feature extraction:* 79 features are extracted from each $256 \times 256 \times 3$ image. These 79 features contain 64 color features, 7 texture features, and 8 shape features.
Step 4. *Feature reduction:* The whole 79 features are reduced via PCA, and the preserved feature standard is to cover at least 95% of the whole variance.
Step 5. The 1653 samples are split into training set (1322) and test set (331) in the proportion of 4:1 randomly. Meanwhile, the training set is divided by stratified 5-fold cross validation.
Step 6. The training set is submitted to train the FNN. The weights/biases of the FNN are adjusted to make the average MSE minimal. FSCABC is the training algorithm.
Step 7. The test dataset is used to analyze the performance of the trained classifier and to calculate the confusion matrix. If acceptable, then output the classifier, otherwise return to Step 6 to re-train FNN.

4. Experiments and results

The experiments were carried out on a P4 IBM platform with Intel Core i3-2330M 2.2 GHz processor and 6 GB RAM running under 64-bit Microsoft Windows 7 operating system. The algorithm was in-house developed on Matlab 2013a (The Mathworks ©) platform. These programs can be run or tested on any computer platforms where Matlab is available.

4.1. Dataset

The fruit dataset was obtained after 6 months of on-site collecting via digital camera and online collecting using http://images.google.com as the main search engine. Split-and-merge algorithm was used to remove the background area; later images were cropped to leave the fruit in the center of the image, and finally downsampled to 256×256 in size.

The data set comprises 1653 images from following 18 different categories: Granny Smith Apples (64), Rome Apples (83), Yellow Bananas (132), Green Plantains (61), Tangerines (112), Hass

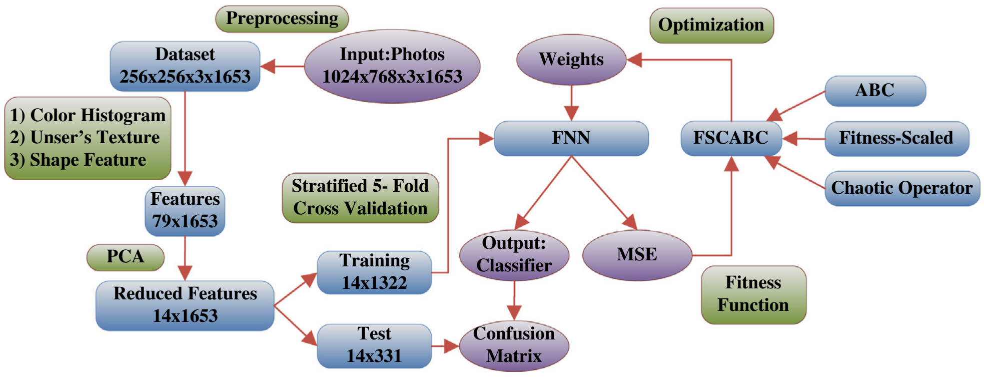


Fig. 9. The flowchart of the proposed fruit recognition system (some numbers in the figure are achieved by following experiments).

Avocados (105), Watermelons (72), Cantaloupes (129), Gold Pineapples (89), Passion Fruits (72), Bosc Pears (88), Anjou Pears (140), Green Grapes (74), Red Grapes (45), Black Grapes (122), Blackberries (97), Blueberries (95), and Strawberries (73). Fig. 10 depicts the samples of different types of fruits in the dataset.

4.2. Feature reduction

The curve of cumulative sum of variance versus number of reduced vectors via PCA is shown in Fig. 11. The detailed data results are listed in Table 4. It (The bold font) shows that merely 14 features can preserve **95.08%** of total variance. The reduced features only cost 17.7% (14/79) of the memory needed for the original 79 features.

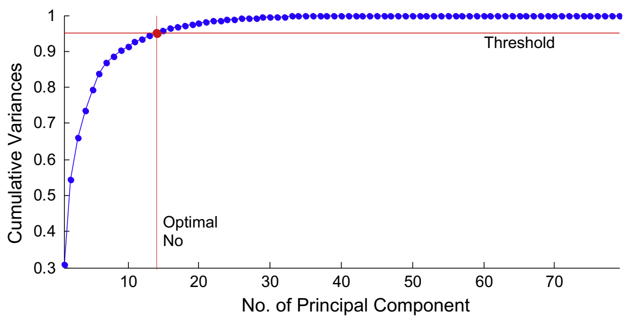


Fig. 11. Feature selection via PCA (threshold is set to 95%).

4.3. Training method comparison

Now that there are 14 reduced features remaining, the structure of the FNN is set to 14-11-18. The input neurons N_I correspond to the number of features as 14. The number of hidden neurons N_H are set to 11 via the information entropy method (Ludwig Jr et al., 2009). The number of output neurons N_O correspond to the number of categories of fruits as 18.

We compared our FSCABC method with latest training algorithm, including BP, momentum BP (MBP), GA, SA, PSO, and ABC. The parameters of these algorithms are obtained using trail-and-error method and listed in Table 5. The maximum iterative epochs are set as 500. Each algorithm ran 20 times on the training set. The

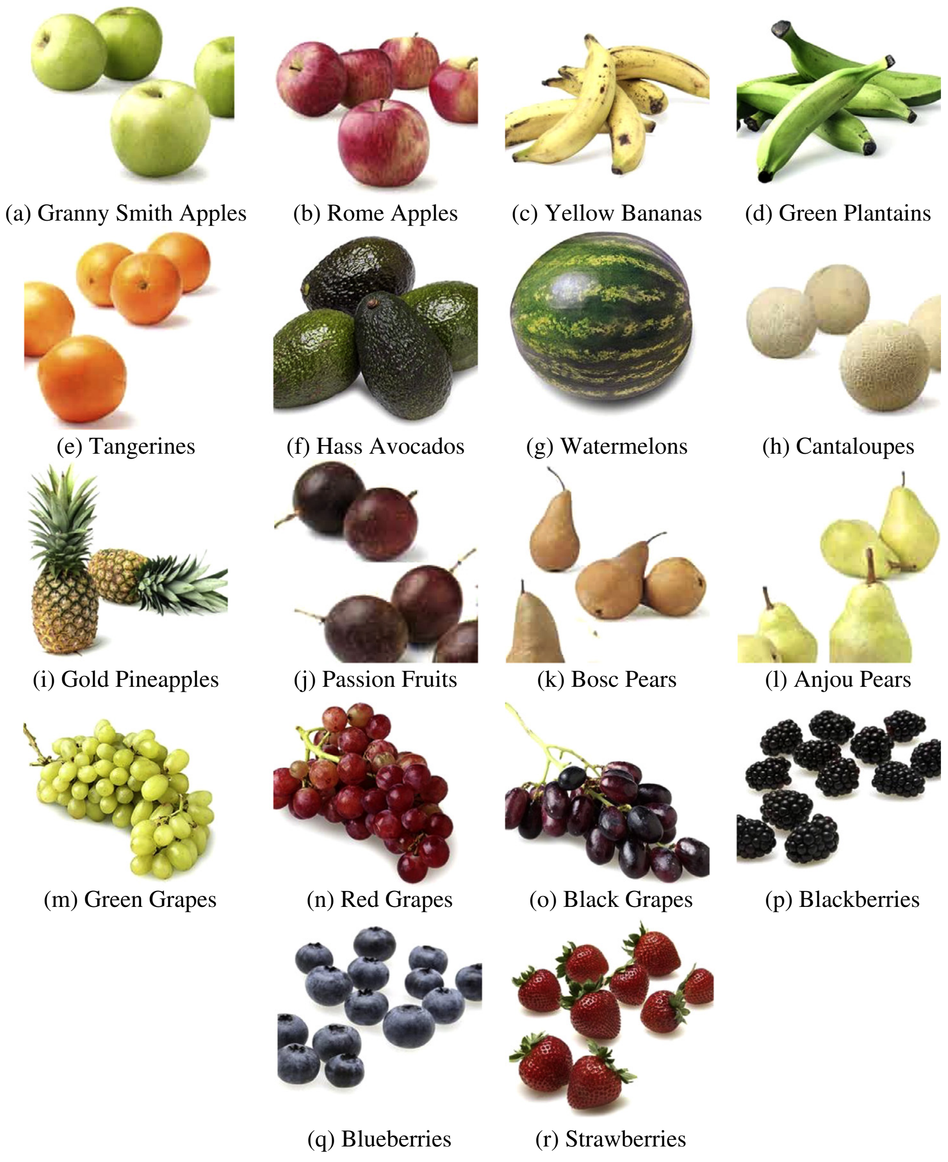


Fig. 10. Samples of *Fruit* dataset of 18 different categories.

Table 4
The cumulative variances of PCA transformed features.

Dimensions	1	2	3	4	5	6	7	8	9	10
Variance (%)	30.73	54.36	66.00	73.60	79.43	83.73	86.82	88.68	90.15	91.45
Dimensions	11	12	13	14	15	16	17	18	19	20
Variance (%)	92.55	93.51	94.35	**95.08**	95.71	96.27	96.76	97.19	97.55	97.87

mean and standard deviation of MSE of all algorithms are listed in Table 6.

4.4. Comparison to other classification methods

In this section, we applied the trained classifier to the test set of 331 samples. We compared our FSCABC–FNN method with GA–FNN, PSO–FNN, ABC–FNN, and kSVM. Table 7 gives the final classification accuracy results.

4.5. Confusion matrix

The confusion matrix of FSCABC–FNN is shown in Fig. 12. Each column of the matrix represents the instances in target class (actual class); meanwhile each row represents the instances in the output class (predicted class). The number in *i*th row and *j*th column represents samples whose target is the *j*th class that was classified as *i*th class. All the misclassification cases are highlighted in yellow.

5. Discussions

Results in Fig. 11 and Table 4 indicate that PCA will accelerate the proposed fruit recognition system remarkably. Although 79 features is not a burden to latest computers, yet after they are reduced to 14 features, we can further accelerate the training and test procedure. Meanwhile removing extra features will enhance the classification accuracy.

Results in Table 6 shows that the worst two algorithms are BP and MBP. The reason is that these two algorithms are designed merely for the linear problem. However, in this fruit classification problem, the output neurons are working in the saturated nonlinear range. The SA does not find the satisfying solutions due to its requirement of large iterative epochs. Conversely, the FSCABC algorithm performs best, obtaining the least mean MSE of 0.0242. The PSO is the second best algorithm with MSE as 0.0267. ABC is the third one with MSE as 0.0310. GA is the fourth one with MSE as 0.0533.

Data in Table 7 shows that the FSCABC–FNN achieves the highest classification accuracy as 89.1%. GA–FNN obtained classification accuracy as 84.8%, PSO–FNN as 87.9%, ABC–FNN as 85.4%, and kSVM as 88.2%. Again, the results on test set demonstrate the superiority of FSCABC–FNN. Besides, it shows that kSVM performed a bit lower of the proposed FSCABC–FNN method as 89.1%. Scholars have pointed out that kSVM exceeds FNN only when the size of dataset is small (Li et al., 2009; Pan et al., 2012). In this study, we created a large-scaled dataset of 1653 samples, and we employed FSCABC to optimize the weights. Hence, the FNN achieved better results than kSVM did.

Checking the bottom line of Fig. 12, we found that 1st class (Granny Smith Apples), 4th class (Green Plantains), 7th class (Watermelons), 12th class (Anjou Pears), 17th class (Blueberries) are all recognized completely correct as 100% classification accuracy. Look back on the pictures in Fig. 10, we found those categories of fruits have distinct features (color, shape, or texture) from others.

Notwithstanding, a few categories of fruits were not recognized so successfully. Here we analyze the worst three categories of fruits. The classification result of the 6th class (Hass Avocados) performs the worst. In the test dataset, there are 21 different pictures of Hass Avocados, however, two of them are misclassified as 2nd class (Rome Apples), one is misclassified as 3rd class (Yellow Bananas), and another two are misclassified as 13th class (Green Grapes), so the rest are recognized correctly leading to a 76.2% (16/21) success rate.

For the 10th class (Passion Fruits), it has 14 samples in the test dataset, one is misclassified as 5th class (Tangerines), and another two are misclassified as 16th class (Blackberries), so it give us a 78.6% (11/14) success rate. For the 14th class (Red Grapes), it has 9 samples in the test dataset, but 1 is misclassified as 5th class (Tangerines) and another is misclassified as 10th class (Passion Fruits), so with the rest 7 samples recognized it gives us a 77.8% (7/9) success rate. In other words, the 6th, 10th, and 14th classes are not clearly distinct by the trained FNN, so their classification accuracies are all lower than 80%. How to recognize those three categories of fruits remains our future work.

Table 5
Parameters of Optimization Algorithms.

Algorithm	Parameter/value
BP	Learning rate = 0.01,
MBP	Learning rate = 0.01, Momentum Constant = 0.9
GA	Population = 20, crossover probability = 0.8, mutation probability = 0.1
SA	Population = 20, initial temperature = 100, final temperature = 0 Temperature decrease function = "exponential"
PSO	Population = 20, maximal velocity = 1, initial weight = 0.5, acceleration coefficient = 1
ABC	Population = 20, food number = 10
FSCABC	Population = 20, food number = 10

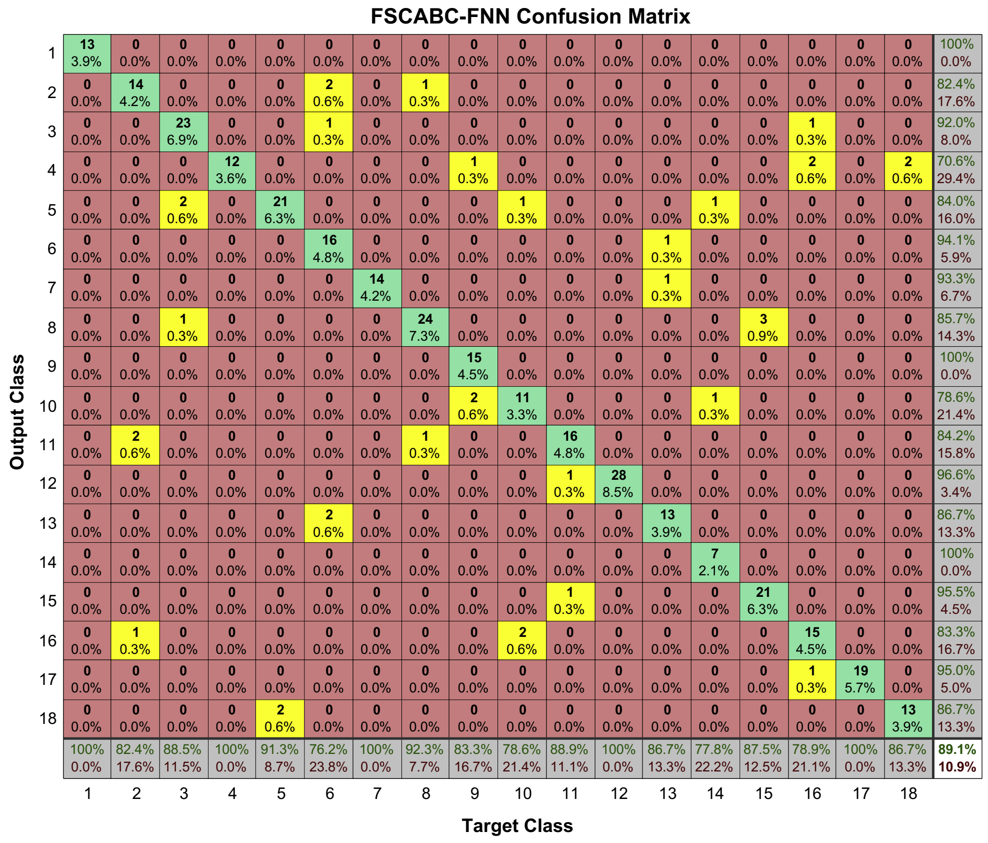
Table 6
Comparison of training algorithms.

Algorithm	Mean MSE	Std. Var. of MSE	Rank
BP	0.2513	0.0333	7
MBP	0.2228	0.0268	6
SA	0.1959	0.0442	5
GA	0.0533	0.0239	4
PSO	0.0267	0.0133	2
ABC	0.0310	0.0176	3
FSCABC	0.0242	0.0135	1

Table 7
Classification accuracy comparison.

Algorithm	Classification accuracy (%)
GA–FNN	84.8
PSO–FNN	87.9
ABC–FNN	85.4
kSVM*	88.2
FSCABC–FNN (our)	89.1

* Result from Zhang and Wu (2012).



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
1	13 3.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	14 4.2%	0 0.0%	0 0.0%	0 0.0%	2 0.6%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	82.4% 17.6%
3	0 0.0%	0 0.0%	23 6.9%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	92.0% 8.0%
4	0 0.0%	0 0.0%	0 0.0%	12 3.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.6%	0 0.0%	2 0.6%	70.6% 29.4%
5	0 0.0%	0 0.0%	2 0.6%	0 0.0%	21 6.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84.0% 16.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 4.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.1% 5.9%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	14 4.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	93.3% 6.7%
8	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	24 7.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.9%	0 0.0%	0 0.0%	0 0.0%	85.7% 14.3%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 4.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.6%	11 3.3%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	78.6% 21.4%
11	0 0.0%	2 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	16 4.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84.2% 15.8%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	28 8.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	96.6% 3.4%
13	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	13 3.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	86.7% 13.3%
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	7 2.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	21 6.3%	0 0.0%	0 0.0%	0 0.0%	95.5% 4.5%
16	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 4.5%	0 0.0%	0 0.0%	83.3% 16.7%
17	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	19 5.7%	0 0.0%	95.0% 5.0%
18	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	13 3.9%	86.7% 13.3%
	100% 0.0%	82.4% 17.6%	88.5% 11.5%	100% 0.0%	91.3% 8.7%	76.2% 23.8%	100% 0.0%	92.3% 7.7%	83.3% 16.7%	78.6% 21.4%	88.9% 11.1%	100% 0.0%	86.7% 13.3%	77.8% 22.2%	87.5% 12.5%	78.9% 21.1%	100% 0.0%	86.7% 13.3%	89.1% 10.9%

Fig. 12. Confusion matrix of FSCABC–FNN with overall classification accuracy of 89.1%.

6. Conclusions and future research

This work proposed a novel classification method based on FSCABC–FNN. We used the combination of color histogram, Unser's texture, and shape features. The experimental results demonstrated that the FSCABC–FNN achieved a significant classification accuracy of 89.1%. It is higher than the results of GA–FNN, PSO–FNN, ABC–FNN, and kSVM.

The contribution of the work lies in the following aspects. (1) We proposed a hybrid feature set, containing color information, texture information, and shape information. (2) We introduced a stratified *K*-fold cross validation to avoid overfitting. (3) We introduced the FSCABC algorithm, and employed it to the training of FNN. (4) We compared the FSCABC with BP, MBP, GA, SA, PSO, and ABC. (5) We compared the FSCABC–FNN with GA–FNN, PSO–FNN, and ABC–FNN. (6) We gave the confusion matrix of the final FSCABC–FNN classifier, indicating which types of fruits the classifier does not work well on.

The future research work will concentrate on following points: (1) extending our research to fruits in severe conditions, such as sliced, dried, canned, tinned, and partially covered fruit**s**; (2) including additional features to increase the classification accuracy; (3) employing cloud computing and other parallel technique to accelerate the algorithm; (4) applying our algorithm in other similar fields as chestnut quality assessment (Donis-González et al., 2013), shrimp color monitoring (Hosseinpour et al., 2013), etc.

Acknowledgments

This work was supported from NNSF of China (No. 610011024) and Nanjing Normal University Research Foundation for Talented Scholars (No. 2013119XGQ0061).

References

Amiri, Z.R., Khandelwal, P., Aruna, B.R., Sahebjamnia, N., 2008. Optimization of process parameters for preparation of synbiotic acidophilus milk via selected probiotics and prebiotics using artificial neural network. J. Biotechnol. 136 (Suppl. 1), S460.

Armand, S., Watelain, E., Roux, E., Mercier, M., Lepoutre, F.-X., 2007. Linking clinical measurements and kinematic gait patterns of toe-walking using fuzzy decision trees. Gait Posture 25 (3), 475–484.

Baltazar, A., Aranda, J.I., González-Aguilar, G., 2008. Bayesian classification of ripening stages of tomato fruit using acoustic impact and colorimeter sensor data. Comput. Electron. Agric. 60 (2), 113–121.

Bolle, R.M., Connell, J.H., Haas, N., Mohan, R., Taubin, G., 1996. VeggieVision: a produce recognition system, In: Applications of Computer Vision, 1996. WACV '96, Proceedings 3rd IEEE Workshop on, Sarasota, FL, USA.

Cano Marchal, P., Martínez Gila, D., Gámez García, J., Gómez Ortega, J., 2013. Expert system based on computer vision to estimate the content of impurities in olive oil samples. J. Food Eng. 119 (2), 220–228.

Coulibaly, P., Evora, N.D., 2007. Comparison of neural network methods for infilling missing daily weather records. J. Hydrol. 341 (1–2), 27–41.

Crone, S.F., Kourentzes, N., 2010. Feature selection for time series prediction – a combined filter and wrapper approach for neural networks. Neurocomputing 73 (10–12), 1923–1936.

Damiand, G., Resch, P., 2003. Split-and-merge algorithms defined on topological maps for 3D image segmentation. Graph. Models 65 (1–3), 149–167.

Donis-González, I.R., Guyer, D.E., Leiva-Valenzuela, G.A., Burns, J., 2013. Assessment of chestnut (Castanea spp.) slice quality using color images. J. Food Eng. 115 (3), 407–414.

Fan, F.H., Ma, Q., Ge, J., Peng, Q.Y., Riley, W.W., Tang, S.Z., 2013. Prediction of texture characteristics from extrusion food surface images using a computer vision system and artificial neural networks. J. Food Eng. 118 (4), 426–433.

Goel, A., Pal, M., 2009. Application of support vector machines in scour prediction on grade-control structures. Eng. Appl. Artif. Intell. 22 (2), 216–223.

Hong, S.G., Maccaroni, M., Figuli, P.J., Pryor, B.M., Belisario, A., 2006. Polyphasic classification of Alternaria isolated from hazelnut and walnut fruit in Europe. Mycol. Res. 110 (11), 1290–1300.

Hosseinpour, S., Rafiee, S., Mohtasebi, S.S., Aghbashlo, M., 2013. Application of computer vision technique for on-line monitoring of shrimp color changes during drying. J. Food Eng. 115 (1), 99–114.

Jackson, J.E., 1991. A User's Guide to Principal Components. John Wiley & Sons.

Jia, X., Yang, X., Cao, K., Zang, Y., Zhang, N., Dai, R., Zhu, X., Tian, J., 2014. Multi-scale local binary pattern with filters for spoof fingerprint detection. Inform. Sci. 268, 91–102.

Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. 8 (1), 687–697.
Karaboga, N., Kalinli, A., Karaboga, D., 2004. Designing digital IIR filters using ant colony optimisation algorithm. Eng. Appl. Artif. Intell. 17 (3), 301–309.
Korsunsky, A.M., Constantinescu, A., 2006. Work of indentation approach to the analysis of hardness and modulus of thin coatings. Mater. Sci. Eng.: A 423 (1–2), 28–35.
Kwak, N., 2008. Principal component analysis based on L1-norm maximization. Pattern Anal. Mach. Intell., IEEE Trans. 30 (9), 1672–1680.
Li, Q., Meng, Q., Cai, J., Yoshino, H., Mochida, A., 2009. Predicting hourly cooling load in the building: a comparison of support vector machine and different artificial neural networks. Energy Convers. Manage. 50 (1), 90–96.
Lipovetsky, S., 2009. PCA and SVD with nonnegative loadings. Pattern Recogn. 42 (1), 68–76.
Llave, Y.A., Hagiwara, T., Sakiyama, T., 2012. Artificial neural network model for prediction of cold spot temperature in retort sterilization of starch-based foods. J. Food Eng. 109 (3), 553–560.
Lou, S., Jiang, X., Scott, P.J., 2012. Algorithms for morphological profile filters and their comparison. Precis. Eng. 36 (3), 414–423.
Ludwig Jr, O., Nunes, U., Araújo, R., Schnitman, L., Lepikson, H.A., 2009. Applications of information theory, genetic algorithms, and neural models to predict oil flow. Commun. Nonlinear Sci. Numer. Simul. 14 (7), 2870–2885.
Maitra, M., Chatterjee, A., 2008. A hybrid cooperative-comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding. Expert Syst. Appl. 34 (2), 1341–1350.
May, R.J., Maier, H.R., Dandy, G.C., 2010. Data splitting for artificial neural networks using SOM-based stratified sampling. Neural Networks 23 (2), 283–294.
Nosaka, R., Fukui, K., 2014. HEp-2 cell classification using rotation invariant co-occurrence among local binary patterns. Pattern Recogn. 47 (7), 2428–2436.
Omid, M., Soltani, M., Dehrouyeh, M.H., Mohtasebi, S.S., Ahmadi, H., 2013. An expert egg grading system based on machine vision and artificial intelligence techniques. J. Food Eng. 118 (1), 70–77.
Ou, X., Pan, W., Xiao, P., 2014. In vivo skin capacitive imaging analysis by using grey level co-occurrence matrix (GLCM). Int. J. Pharm. 460 (1–2), 28–32.
Pan, S., Iplikci, S., Warwick, K., Aziz, T.Z., 2012. Parkinson's disease tremor classification – a comparison between support vector machines and neural networks. Expert Syst. Appl. 39 (12), 10764–10771.
Pennington, J.A.T., Fisher, R.A., 2009. Classification of fruits and vegetables. J. Food Compos. Anal. 22 (Suppl. 1), S23–S31.
Pereira, A.C., Reis, M.S., Saraiva, P.M., Marques, J.C., 2011. Madeira wine ageing prediction based on different analytical techniques: UV–vis, GC-MS, HPLC-DAD. Chemom. Intell. Lab. Syst. 105 (1), 43–55.
Pholpho, T., Pathaveerat, S., Sirisomboon, P., 2011. Classification of longan fruit bruising using visible spectroscopy. J. Food Eng. 104 (1), 169–172.
Poursamad, A., 2009. Adaptive feedback linearization control of antilock braking systems using neural networks. Mechatronics 19 (5), 767–773.
Raheja, J.L., Kumar, S., Chaudhary, A., 2013. Fabric defect detection based on GLCM and Gabor filter: a comparison. Opt. – Int. J. Light Electron Opt. 124 (23), 6469–6474.
Rocha, A., Hauagge, D.C., Wainer, J., Goldenstein, S., 2010. Automatic fruit and vegetable classification from images. Comput. Electron. Agric. 70 (1), 96–104.
Sakthivel, N.R., Sugumaran, V., Babudevasenapati, S., 2010. Vibration based fault diagnosis of monoblock centrifugal pump using decision tree. Expert Syst. Appl. 37 (6), 4040–4049.
Seng, W.C., Mirisaee, S.H., 2009. A New Method for Fruits Recognition System. In: Electrical Engineering and Informatics, 2009. ICEEI '09. International Conference on, Selangor.
Siang Tan, K., Mat Isa, N.A., 2011. Color image segmentation using histogram thresholding – fuzzy C-means hybrid approach. Pattern Recogn. 44 (1), 1–15.
Singh, N., Sinha, A., 2010. Chaos-based secure communication system using logistic map. Opt. Lasers Eng. 48 (3), 398–404.
Unser, M., 1995. Texture classification and segmentation using wavelet frames. Image Process., IEEE Trans. 4 (11), 1549–1560.
Wang, Y., Li, B., Weise, T., 2010. Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. Inform. Sci. 180 (12), 2405–2420.
Xiao, Y., Jia Zou, J., Yan, H., 2001. An adaptive split-and-merge method for binary image contour data compression. Pattern Recogn. Lett. 22 (3–4), 299–307.
Zanaganeh, M., Mousavi, S.J., Etemad Shahidi, A.F., 2009. A hybrid genetic algorithm-adaptive network-based fuzzy inference system in prediction of wave parameters. Eng. Appl. Artif. Intell. 22 (8), 1194–1202.
Zhang, Y., Wu, L., 2011. Crop classification by forward neural network with adaptive chaotic particle swarm optimization. Sensors 11 (5), 4721–4743.
Zhang, Y., Wu, L., 2012. Classification of fruits using computer vision and a multiclass support vector machine. Sensors 12 (9), 12489–12505.
Zhang, Y., Wu, L., Wang, S., 2011a. Magnetic resonance brain image classification by an improved artificial bee colony algorithm. Progress Electromagn. Res. 116, 65–79.
Zhang, Y., Wu, L., Wang, S., 2011b. UCAV path planning based on FSCABC. Inform. – Int. Interdiscipl. J. 14 (3), 687–692.
Zhang, Y., Wu, L., Wang, S., 2013. UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization. Math. Problems Eng. 2013, 1–9.
Zhang, Y., Wu, L., Wei, G., Wang, S., 2011c. A novel algorithm for all pairs shortest path problem based on matrix multiplication and pulse coupled neural network. Digit. Signal Process. 21 (4), 517–521.