TotoBee: Tomato Maturity Grading using Artificial Bee Colony Algorithm and Artificial Neural Networks

Presented to the Faculty of the

Natural Sciences and Mathematics Division

University of the Philippines Visayas

Tacloban College

In Partial Fulfillment of the Requirements

for the Degree of

Bachelor of Science in Computer Science

Presented By:

Harvey Jake G. Opeña

2007-24477

April 2013

Approval Sheet

This project entitled “TotoBee: Tomato Maturity Grading using Artificial Bee Colony Algorithm and Artificial Neural Networks” by Harvey Jake G. Opeña was presented to the Faculty of the Natural Sciences and Mathematics Division of the University of the Philippines Visayas Tacloban College.

This project is hereby accepted and approved in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

John Paul T. Yusiong

Adviser

Date

Acknowledgement

This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. I would like to take this opportunity to express my deepest gratitude to the people who have been instrumental in the successful completion of this paper.

To my adviser, Prof. John Paul T. Yusiong, my deepest thanks for his tremendous support and help, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me during the time of doing research and writing of this thesis. Without his encouragement and guidance this thesis paper would not have materialized.

To Mr. Turibio Gabornes and Mr. Joselito Egargo for accommodating me in the Demonstration Farm of the Local Government Unit (LGU) of Lawaan, Eastern Samar during the verification of my test samples needed for my thesis paper, and also for the cooperation during my visit.

To my classmates and friends, my utmost gratitude to all of you for the sincerity and encouragement that I will never forget. These people have been one of my inspirations as I hurdle all the obstacles in the completion this research work.

To my family for the cooperation and encouragement which helped me in the completion of this project. Also, for their guidance and constant supervision as well as for providing necessary information regarding the research and also for their financial support.

To all the other people who in one way or another participated in the success of this paper, your efforts are very much appreciated.

Last but not the least, the one above all of us, the omnipresent God, for answering our prayers for giving me the strength despite our constitution wanting to give up, thank you so much Dear Lord.

Abstract

Tomato maturity grading is very essential in commercial farms that produce large quantities of tomatoes every harvest time. Tomato grading is usually done by human graders by matching the surface color of tomatoes to the United States Department of Agriculture's (USDA) tomato color chart which shows six maturity stages: Green, Breakers, Turning, Pink, Light Red, and Red. But due to some uncontrolled factors, manual classification involving human graders is very prone to misclassification. Thus, this paper introduces an automated tomato classification system that uses an Artificial Neural Network (ANN) classifier trained by the Artificial Bee Colony algorithm (ABC). In order to effectively classify tomatoes, we combine five color features (Red, Green, Red-Green, Hue, a*) from three color models (RGB, HSI, CIE L*a*b*). These features are used as inputs to the ANN classifier. Experiment results show that the ABC-trained ANN classifier performed well in tomato classification and achieved a high classification accuracy rate. Also, results show that combining the color features produced better accuracy rate than using a single color model. With these results, an automated tomato classification system using an ABC-trained ANN classifier can be used to replace the manual classification procedure as it minimizes the chances of misclassification.

Keywords: Tomato Grading, Artificial Bee Colony Algorithm, ABC, Artificial Neural Networks, Image Processing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Tomato is one of the most important and popular fruit in the world which can be consumed in several ways as an ingredient in many dishes, sauces, and drinks. In fact, tomato production averages 150 million tons each year with a total value of $74.1 billion [23]. Commercial farms are the major contributors in the production of tomatoes all over the world.

Generally, fruits such as tomato are sorted and grouped into several classes before shipping to the market. This task is manually done using human labor. Farmers (human graders) in most commercial farms classify tomato maturity by comparing its surface color to the USDA color chart [19]. Misclassification can always happen with this kind of scheme. Thus, an automated classification of tomatoes using computer vision and image processing techniques will increase the accuracy rate and improve the classification performance.

Image classification has been quite successful using statistical methods. With the fast advancement in data mining, a new field of study has appeared – image mining. In effect, many research groups have gained interest in the field of image classification using image mining techniques. Perhaps, one of the most common techniques in image classification is based on Artificial Neural Network (ANN) [5].

ANN's success greatly depends on the algorithm used in training. In 2005, Artificial Bee Colony Algorithm, a swarm based algorithm, was developed by Dervis Karaboga. In the works of Shah, et. al [17] and Kumbhar, et. al [30], ABC has been confirmed to be an efficient trainer of ANNs.

The increasing popularity of image mining and the existence of great image classification algorithms have encouraged many researchers to work with tomato classification based on quality or ripeness. In this study, an ABC-trained ANN will be evaluated on how well it performs when applied to the tomato classification problem.

1.1 Tomato

Tomatoes, scientifically known as *Lycopersicon esculentum*, are said to have been first cultivated by the Aztec civilization in central Mexico around 700 A.D. The word "tomato" actually comes from the Aztecan word "tomatl" which means "the swelling fruit". It has several varieties which differ in shape, size, and color [39].

Although tomato is a fruit, it does not have the same quality of sweetness of other dessert fruits. In effect, it is sometimes regarded as a part of the vegetable family. Tomato is mainly consumed as an ingredient in many dishes and commercial products instead of eating it raw. Specifically, tomatoes are used in the production of many different sauces, tomato pastes, and ketchups.

A tomato, depending on its variety, fully matures within 55 days to 85 days on average [20]. This maturity process starts from green to red coloration of its surface. Figure 1.1 shows the color variations between half-ripe and ripe stages of tomatoes.

Taken from [27]

Figure 1.1: Ripe tomatoes

1.1.1 Tomato Ripening Stages

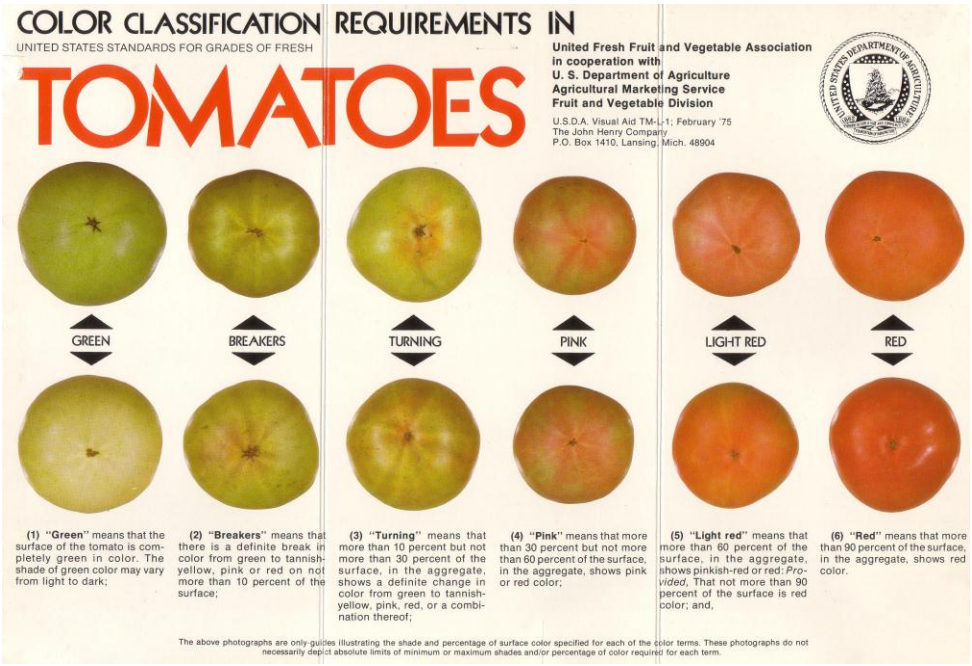
Basically, the surface color of tomato is the most observable characteristic in classifying its maturity or ripeness. In 1991, The United States Department of Agriculture (USDA) created a color chart as a standard for the purpose of tomato classification. According to

the USDA, tomato ripeness is divided into six stages. Table 1.1 summarizes the color percentages of the different stages.

Table 1.1: USDA tomato ripening stages color distribution

Stage	Coloration
Green	The surface is completely green.
Breaker	10% or less of the surface is not green.
Turning	10 - 30% of the surface is not green.
Pink	30 – 60% of the surface is not green.
Light Red	60 – 90% of the surface is not green.
Red	90% - 100% of the surface is not green.

Figure 1.2 shows the different ripening stages of a tomato. It is obvious that as the tomato matures, the total green coloration on the surface starts to decrease while the total red coloration starts to increase.



Taken from [40]

Figure 1.2: USDA tomato ripening stages

1.1.2 Tomato Classification

The traditional way of tomato ripeness classification is very tiring and prone to error. Human graders can suffer from visual fatigue and stress that may affect the quality of grading. Also, human grading may vary from one person to another [12]. Considering these factors, there can be a high percentage of error in classification. Thus, there is a need to automate the classification of tomatoes using a tool that has high reliability and accuracy.

Automated tomato classification is the process of sorting tomatoes into different classes by means of machine vision and image processing techniques. Basically, images of tomatoes are taken as inputs in the classification process. From the images, certain features are extracted and further analyzed.

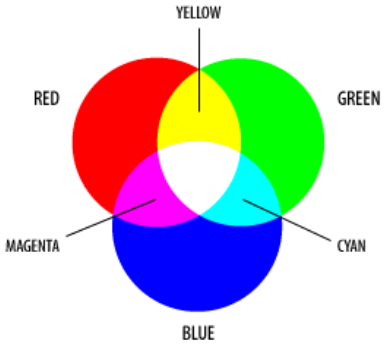
In tomato classification, the type of image used can affect its result because different features are fed to the classifier. Spectral images were used in the work of Polder, et. al [22, 23]. While Zhang, et. al [32] used Magnetic Resonance Imaging (MRI) which is usually used in medical operations. The most common type of image used is the grayscale or the standard RGB image.

1.2 Color Space

Color space, also known as color model, is a method of specifying a color numerically or mathematically. It is usually represented in three-dimensional space of color because of the fact that human eyes perceive colors through three receptors called cones. Each of these cones corresponds to the primary colors of light namely: red, green and blue [8][11]. In tomato classification, the RGB, HSI and CIE L*a*b* color models are the most commonly used.

RGB is composed of three color components (red, green and blue) which can be mixed in various proportions to produce all other colors. It is the most widely used color space in computer graphics because of the simplicity of its architecture unlike other color spaces. Also, most digital devices use the RGB format to output images. Although RGB

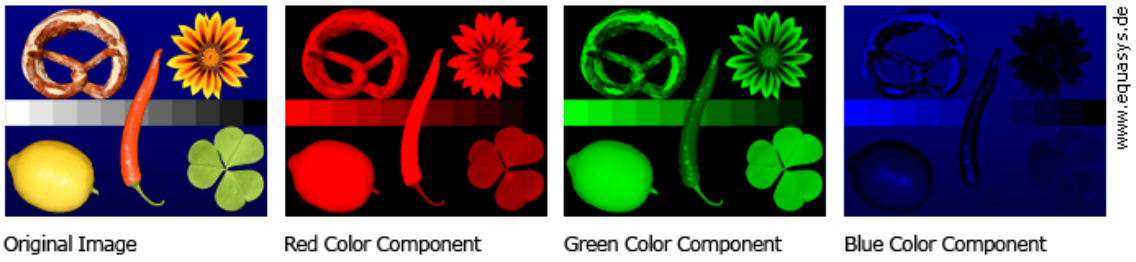
gains the most popularity, it is not the most efficient when dealing with real world images [38].



Taken from [38]

Figure 1.3: RGB color model

In digital imaging, an RGB image is basically a channel sandwich of three channels: red, green, and blue. So when these channels are combined, it produces the RGB image [36]. Figure 1.4 shows the three channel composition of an RGB image.



Taken from [18]

Figure 1.4: RGB channels

HSI color space represents color in three components: Hue (*H*), Saturation (*S*), and Intensity (*I*). Hue component defines the color ranging from 0 degrees to 360 degrees. Saturation defines how strong or weak the color is and ranges from 0 to 100%. Lastly, the Intensity component defines the brightness also ranging from 0 to 100% [21].

RGB to HSI Conversion

$$H = \cos^{-1} \frac{(0.5*(2R-G-B))}{\sqrt{(R-G)^2-(R-G)*(G-B)}} \quad \textbf{(1)}$$

$$S = 1 - 3 * \min(R, G, B) \quad \textbf{(2)}$$

$$I = \frac{(R + G + B)}{3} \quad \textbf{(3)}$$

In 1976, the *Comission Internationale de l'Eclairage* (International Commission on Illumination) created CIE L*a*b (CIELAB). The L* component closely matches how human eye perceives lightness and this runs from white to black color. On the other hand, a* component runs from red to green while b* component runs from yellow to blue. Unlike RGB, CIE L*a*b* is device independent [10].

RGB to CIE L*a*b* conversion is a complex process. It involves two steps: first is converting RGB to CIE XYZ and then CIE XYZ to CIELAB.

RGB to CIE L*a*b* (CIELAB) Conversion

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{pmatrix} \begin{pmatrix} R_{linear} \\ G_{linear} \\ B_{linear} \end{pmatrix} \quad \textbf{(4)}$$

$$L^* = 116 f(Y/Y_n) - 16 \quad \textbf{(5)}$$

$$a^* = 500[f(X/X_n) - f(Y/Y_n)] \quad \textbf{(6)}$$

$$b^* = 200[f(Y/Y_n) - f(Z/Z_n)] \quad \textbf{(7)}$$

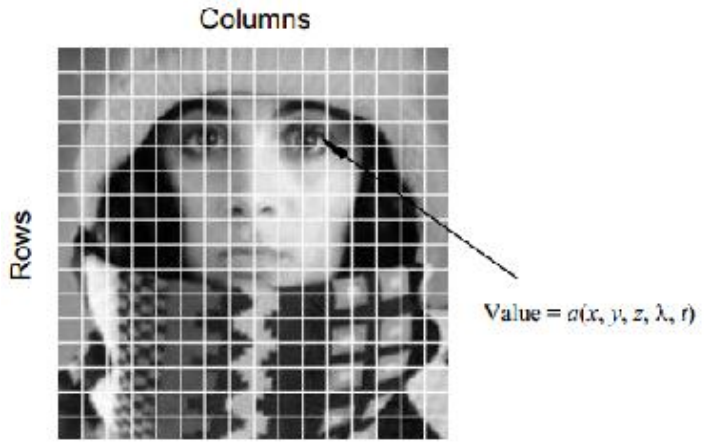
where

$$f(t) = \begin{cases} t^{1/3}, & t > (6/29)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29}, & otherwise \end{cases} \quad \textbf{(8)}$$

1.3 Image Processing

Basically, image processing is the method of converting an image into a more meaningful image through some manipulations and operations. It takes a two-dimensional image as an input and modifies it using a set of parameters. The output of the process can be a pure image or a set of characteristics of that image. Image processing is mainly used for the purpose of visualization, image restoration, image retrieval, and image recognition [26].

In the past few decades, image processing has been a challenging domain in computer programming. Since it has to deal with images as inputs, it requires a computer system that can manage graphical operations and has adequate memory capacity. With the fast advancement of technology, image processing has become a more interesting field [2]. It is widely used in remote sensing, medical treatment, film industry, and printing industry.

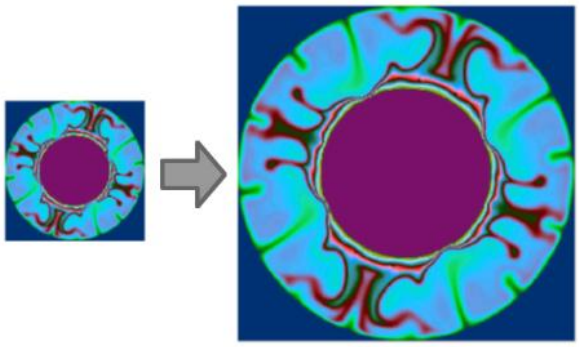


Taken from [13]

Figure 1.5: Image processing: Digitization of an image

1.3.1 Resizing

Image resizing, or scaling, is a technique that supports image analysis by either expanding or shrinking an image. Expanding or magnifying involves interpolating new values based on the original image to come up with additional pixels making the output image larger than the original. On the other hand, shrinking reduces the size of the image by resampling the pixels to produce lesser number of pixels [24]. Figure 1.6 shows an example of image magnification.



Taken from [23]

Figure 1.6: Original image and magnified image

1.3.2 Background Removal

In some image classification tasks, it is necessary to remove some parts of an image to extract meaningful information such as the foreground object, also called the region of interest (ROI). The region of interest is the part of an image that researchers are interested in and is the focus of image processing and image analysis. The area in the image that is not part of the ROI is called the background which needs to be removed [31].

1.3.2.1 Grayscale

An image whose pixel values only represent its intensity or brightness is called a grayscale image (also called black-and-white). In an RGB grayscale image, the three color components (red, green, and blue) have equal values. The pixel values range from white color (total transmission or reflection of light) to black color (total absence of transmitted or reflected light). In effect, images of this type are composed of exclusively shades of gray with values ranging from 0-255 [18].

In image processing, grayscale conversion is a common preliminary method. There are several ways to convert a color image to grayscale image. Some of which are listed below [37].

- Averaging

$$Gray = (Red + Green + Blue)/3 \quad \textbf{(9)}$$

- Luminance

$$Gray = (Red * 0.299 + Green * 0.587 + Blue * 0.114) \quad \textbf{(10)}$$

- Desaturation

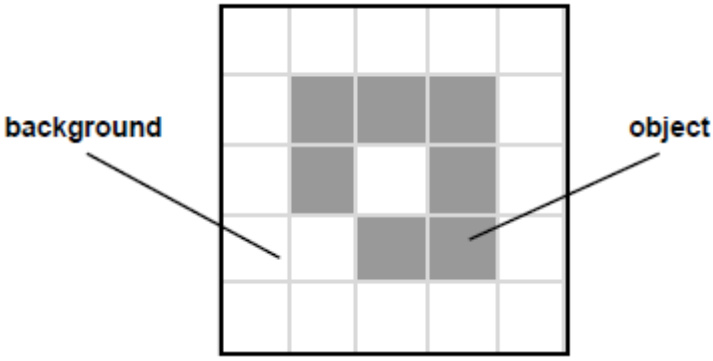
$$Gray = (Max(Red, Green, Blue) + Min(Red, Green, Blue))/2 \quad \textbf{(11)}$$

Taken from [37]

Figure 1.7: A color image converted to grayscale

1.3.2.2 Binary Mask

Binary image is an image whose pixel values can only be 1 or 0, hence it is called binary image. To binarize an image, a grayscale image is taken as an input converting each of its pixel value to either 1 or 0. Most of the time, pixels having the value 1 are represented by white color while pixels having the value 0 are represented by black color. Binary image can be used as a mask to separate the foreground object/s from its background [29, 51]. Figure 1.8 shows a pixel-level representation of a binary mask. Gray and white colors are used in Figure 1.8 to illustrate the object and background pixels.



Taken from [6]

Figure 1.8: Pixel-level representation of a binary mask

Taken from [35]

Figure 1.9: (a) Grayscale image. (b) Binary mask of (a).

1.3.2.2.1 Otsu Thresholding

Otsu's method [35], developed by Nobuyuki Otsu, is a binarization algorithm based on histogram. It assumes that the image has two classes of pixels (i.e. foreground and background) or bi-modal histogram. This method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold. The goal is to find a threshold that minimizes within-class variance defined as a weighted sum of variances of the two classes shown in equation (12).

$$\sigma^2_{Within}(T) = \omega_0(T)\sigma_0^2 + \omega_1(T)\sigma_1^2(T) \quad \textbf{(12)}$$

Instead of minimizing the within-class variance, it can be expressed as a maximization problem of the between-class variance shown in equations (13) and (14). This approach is far simpler and quicker than the previous one.

$$\sigma^2_{Between}(T) = \sigma^2 - \sigma^2_{Within}(T) \quad \textbf{(13)}$$

$$\sigma^2_{Between}(T) = \omega_0(T)\omega_1(T)[\mu_0(T) - \mu_1(T)]^2 \quad \textbf{(14)}$$

where T is the threshold value; ω_0 and ω_1 are the class probabilities; μ_0 and μ_1 are the class means. The class probabilities can be computed using equations (15) and (16) while the class means can be computed using equations (17) and (18).

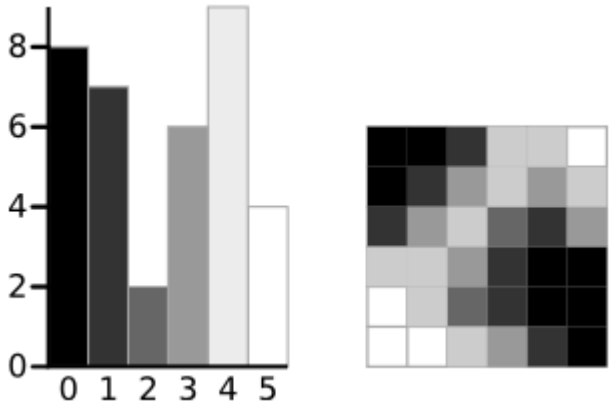
$$\omega_0(T) = \sum_{i=0}^{T} p(i) \quad \textbf{(15)}$$

$$\omega_1(T) = \sum_{i=T+1}^{L} p(i) \quad \textbf{(16)}$$

$$\mu_0(T) = \frac{\sum_{i=0}^{T} p(i)*i}{\omega_0(T)} \quad \textbf{(17)}$$

$$\mu_1(T) = \frac{\sum_{i=T+1}^{L} p(i)*i}{\omega_{1(T)}} \quad \textbf{(18)}$$

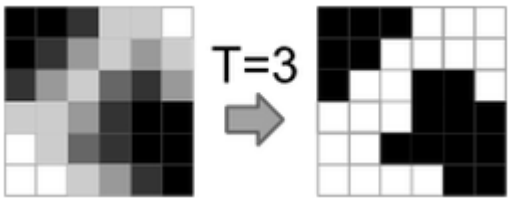
To illustrate Otsu thresholding, a 6 x 6 image is shown in Figure 1.10. The image is a 6-level grayscale containing a total of 6 shades of gray. The graph on the left is the corresponding histogram in which the x-axis represents the range of gray colors (indexed 0-5, depending on the brightness) in the image and the y-axis represents the total pixel count a specific color occupies in the image.



Taken from [35]

Figure 1.10: 6-level grayscale (right) and its histogram (left)

In this example, Otsu method can iterate up to 6 times trying to find the optimal threshold. In the 4th iteration, where T = 3, the within-class variance is at minimum and at the same time the between-class variance it at maximum. The resulting binary image is shown in Figure 1.11.

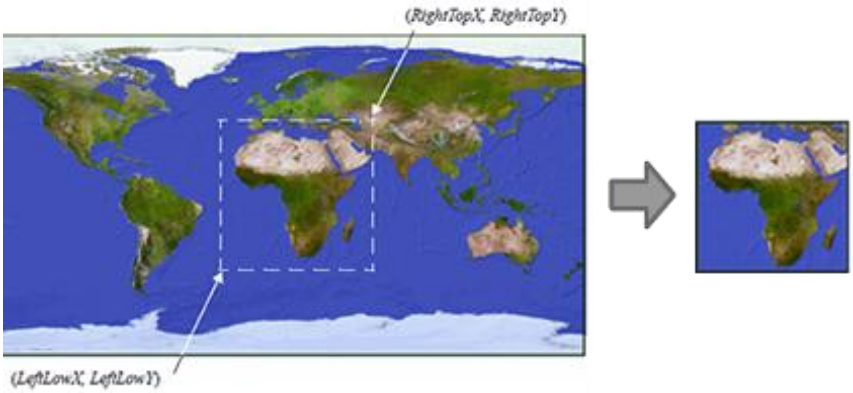


Taken from [35]

Figure 1.11: Result of Otsu thresholding

1.3.3 Cropping

Image cropping is a very useful preprocessing method in image analysis. It is the extraction of a rectangular region of interest from an image. The purpose of image cropping is to remove the regions that contain less useful information or of no use at all. [24]. Figure 1.12 shows an example of how a rectangular region is cropped from an image.



Taken from [24]

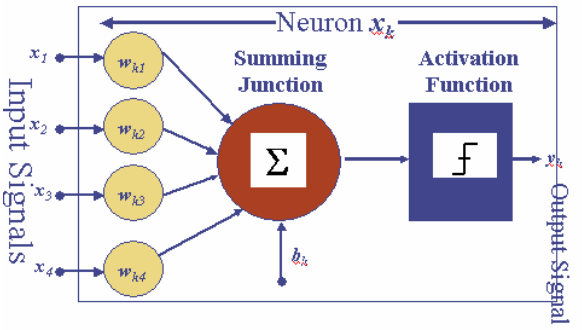
Figure 1.12: Original image and cropped image

1.4 Artificial Neural Networks

In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts investigated how the neurons inside the brain work. They designed the very first artificial neural network using simple circuits. After that, many researches were conducted to improve the artificial neural network [34].

An artificial neural network (ANN) is an emulation of the biological neural system. Just like the human brain, ANN learns by example. Neurons are represented by artificial neurons arranged in layers connected by synaptic weights. The major advantages of this algorithm include the ability to solve non-linear problems, parallel systems, ability to learn, and it can be implemented in any application [25].

In terms of information flow, artificial neural networks are classified into two namely: feed-forward and recurrent neural networks. In a feed-forward neural network the information can only flow forward, while in a recurrent neural network the information can flow forward and backward [25]. Moreover, artificial neural networks can be divided according to the number of layers. They are the single-layer perceptron and the multilayer perceptron.



Taken from [22]

Figure 1.13: Basic structure of a single neuron

1.4.1 Activation Functions

An activation function is a mathematical equation used in artificial neural networks. To obtain the value of an output node in a neural network, the weighted sum of the input nodes is fed to the activation function. Choosing an activation function can be critical; it depends on the problem being solved [14]. The most common functions are given below:

- Linear Function

$$f(x) = x \tag{19}$$

- Piecewise Linear Function

$$f(x) = \begin{cases} 1 \ if \ x \geq \frac{1}{2} \\ x \ if -\frac{1}{2} > x < \frac{1}{2} \\ 0 \ if \ x \leq \frac{1}{2} \end{cases} \tag{20}$$

- Threshold Function

$$f(x) = \begin{cases} 1 \ if \ x \geq 0 \\ 0 \ if \ x < 0 \end{cases} \tag{21}$$

- Sigmoid Function

$$f(x) = \frac{1}{1+e^{-x}} \tag{22}$$

1.4.2 Mean Square Error

The Mean Square Error (MSE) is defined as the average of the square of the error. In neural networks, the error is computed as the difference between the target value and the actual value [17]. Shown below is the equation for MSE:

$$MSE = \frac{1}{N*M}\sum_{i=1}^{N}\sum_{j=1}^{M}(T_{ij} - A_{ij})^2 \quad \textbf{(23)}$$

where N is the number of pattern. M is the number of output nodes. T is the target value and A is the actual computed value of pattern i and node j.

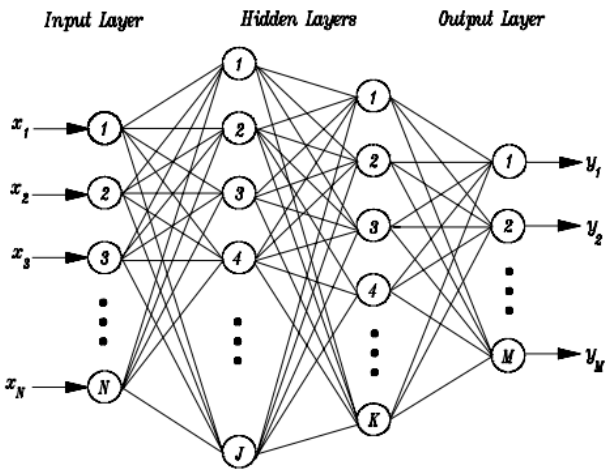
1.4.3 Feed-forward Multilayer Perceptron

In 1958, Rosenblatt conceptualized the perceptron model which is the foundation of many types of artificial neural networks. A perceptron is defined as follows [7]:

"*A perceptron is a device which computes a weighted sum of its inputs, and puts this sum through a special function, called the activation, to produce the output. The activation function can be linear or nonlinear."*

A linear perceptron cannot solve problems that are not linearly separable like the XOR problem. To overcome this limitation of the linear perceptron, a nonlinear perceptron layer called the hidden layer can be added to the network. One of the neural networks with this type of architecture is the multilayer perceptron (MLP) [7].

MLPs [33] are feed-forward and fully-connected networks. From the input to the output layers, every node is connected to all nodes in the following layer making it fully-connected. In addition, the flow of information is unidirectional starting from the input layer to the hidden layer/s and finally to the output layer. Figure 1.14 shows the structure of a feed-forward multilayer perceptron.



Taken from [14]

Figure 1.14: The multi-layer perceptron network

1.5 Artificial Bee Colony Algorithm

In 2005, Dervis Karaboga introduced a swarm-based modern meta-heuristic algorithm which models the intelligent foraging behavior of honey bees. ABC algorithm combines local search with global search in order to balance the exploration and exploitation processes. Also, the ABC algorithm is very simple because it uses few control parameters. In addition, it can handle both constrained and unconstrained problems [3, 4].

Taken from [28]

Figure 1.15: Honey bees

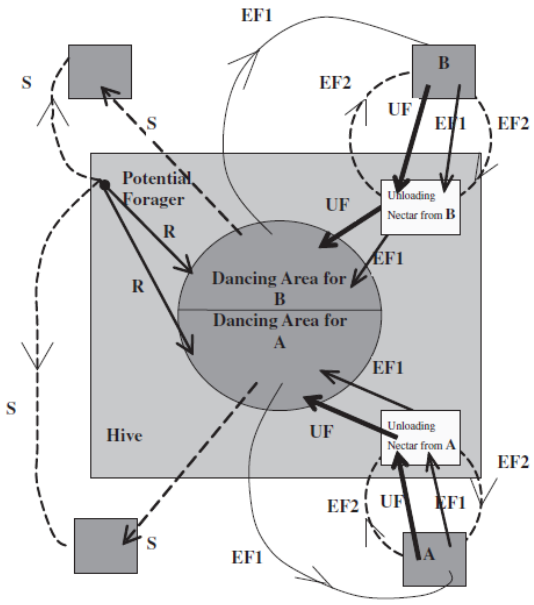
1.5.1 Swarm Intelligence

The concept of Swarm Intelligence (SI) is inspired by the collective behavior of many social insect colonies interacting in their natural surroundings. Typically, these insects are ants, termites, birds, wasps and bees. Self-organization and division of labor are the two sufficient characteristics of an intelligent swarm. A swarm self-organizes when each individual interacts with each other locally. On the other hand, the concept of division of labor in a swarm means assigning the different tasks into specialized and cooperating individuals making the performance more efficient [28].

1.5.2 ABC Components

There are three essential components involved in the ABC model: employed and unemployed foraging bees and food sources. Unemployed bees have two types: onlooker and scout bees. All of these bees search for their food sources around the hive [28].

1. **Food sources:** A food source is a representation of a solution in ABC. The nectar amount of a food source is the basis of its profitability. The higher the nectar value of a food source, the more likely it will be chosen by a bee.
2. **Employed bees:** Every employed bee is associated with a specific food source. These bees carry information about the food sources in which they are "employed" to and share the information to the onlooker bees.
3. **Unemployed bees:** These are bees that are still looking for food sources for "employment". Onlooker and scouts bees are the two types of unemployed bees. Onlooker bees look for food source according to the information shared by the employed bees. While the scout bees are formerly employed bees that cannot find better food sources, thus they abandon their food sources and randomly look for another.



Taken from [28]

Figure 1.16: Behavior of honey bees foraging for nectar

In Figure 1.16, there are two food sources, A and B. EF1 and EF2 are the employed foragers which first gather information about the food sources. When an employed forager abandons a food source, it becomes an unemployed forager (UF) and returns back to the dancing area to look for another food source [28].

1.5.3 ABC Pseudocode

The detailed pseudocode of ABC algorithm is show below:

[1] Initialize the population of solutions x_{ij}

[2] Evaluate the population

[3] Set cycle = 1

[4] **Repeat**

[5] Produce new food source positions v_{ij} in the neighborhood of x_{ij}

[6] Evaluate the population

[7] Apply the greedy selection between x_i and v_i

[8] Calculate the probability P_i

[9] Normalize Pi value into [0, 1]

[10] Produce new positions v_i from x_i depending on P_i

[11] Evaluate the population

[12] Apply the greedy selection between x_i and v_i

[13] Determine the abandoned solution, if it exists

[14] Replace it with a new solution x_i

[15] Memorize the best food source position

[16] Set cycle = cycle + 1

[17] **Until** cycle = Maximum Cycle Number (MCN)

During the employed bees phase, the bees find new food source positions using the following equation:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{hj}) \quad \textbf{(24)}$$

The symbol ϕ_{ij} is a random number between [-α,α] and x_h is a randomly selected food source. After producing the food source v_{ij}, its fitness is compared with the fitness of the current food x_{ij}. A greedy selection is performed between the two food sources [3]. The fitness function is shown in equation (25).

$$fit_i = \frac{1}{1+f_i} \quad \textbf{(25)}$$

Using the fitness values of the employed bees, the onlooker bees can now choose their respective food sources defined by the probability function in equation (26) [3].

$$P_i = \frac{fit_i(x_{ij})}{\sum_{m=1}^{SN} fit_i(x_{ij})} \quad \textbf{(26)}$$

Chapter 2

Literature Review

Image classification has become a very interesting area of study in computer vision due to its various applications such as in remote sensing, medicine, and agriculture. With the fast advancements of image processing techniques, research studies on image classification have rapidly increased over the years. In agriculture, fruit quality or maturity classification is a very popular choice of study.

In the study conducted by Gejima, et. al [16], tomato maturity was used to judge its quality. Two color models, RGB and CIE L*a*b*, were used and compared. To find out the differences among various kinds of tomatoes, a histogram was created for each color model and the correlation lines were obtained. It was analyzed that the pixel count of G(36) in RGB and average value of a* component in CIE L*a*b* are important factors in maturity. But when compared, the average value a* had a higher relation with maturity. Thus, a* was used to create a suggestion table for tomato maturity classification. During the testing phase, 50 samples were used and the result shows that 48 out of 50 were correctly classified.

Using three color models (RGB, HSV, and CIE L*a*b*), Nagata, et. al estimated the tomato in five ripening stages [15]. Nine color images were taken for each maturity stage, with a total of 45 tomato images, using a CCD camera. The effectiveness of the three color models were compared by analyzing the features from the each color model histogram. The well-known linear discriminant analysis (LDA) was used as the statistical classification model. During the model development, 80-100% of the samples were correctly classified. But it did not perform well in the verification phase with success rates of 62.5% (HSV), 60.0% (CIE L*a*b*), and only 35.0% (RGB). HSV and CIE L*a*b got higher success rates compared to RGB because of their invariance with the lighting level. Moreover, it was concluded that using a single component for each color model (hue component in HSV, a* component in L*a*b*, and red component in RGB) can be effective to discriminate ripening classes of tomatoes.

A rapid tomato quality sorting in real-time processing is introduced by Baek, et. al [4]. The quality of tomatoes was estimated with respect to their color or the level of maturity. Three maturity stages were defined: immature, half-ripe, and ripe. In order to evaluate the level of maturity, RGB images were taken and converted into CIE L*a*b* color space. The components a* and b* were extracted and used for classification. To decide the optimal ripening stage of tomatoes, two optimal boundary equations f_1 (a*, b*) and f_2 (a*, b*) were formulated using a* and b* components. If f_1 (a*, b*) > 0, the tomato is immature. If f_2 (a*, b*) < 0, the tomato is ripe. And if f_1 (a*, b*) < 0 > f_2 (a*, b*), the tomato is half-ripe. Experimental results show that the success rates were 85.19% during the calibration and 80.56% during the validation.

Wang, et. al [19] applied a vision-based judgment in tomato maturity classification. In the study, tomatoes were divided into five maturity stages: breakers, turning, pink, light-red, and red. Tomato images were taken under natural illumination and growth conditions. During tomato acquisition, a white paper was placed at the back of the target tomato to make the image preprocessing easier. Two methods were applied in the experiment: using near-infrared images and using the RGB/HSI images. The segmentation of tomatoes was done using the Otsu's method. In the image analysis, the average intensity of near-infrared image cannot be used to judge directly the maturity of tomatoes under all stages. Using the HSI model, the average of hue component changes close to linear and thus, it can separate all five stages. The distinguishing middle points of the hue average as tomato matures are 43, 33, 23, and 16. Likewise, R-G color difference under the RGB color model can distinguish the middle points of all stages. The distinguishing middle points are 0, 23.5, 42.5, and 70.

Zhang and McCarthy investigated the potential of magnetic resonance imaging (MRI) in classifying tomatoes into different maturity stages [32]. MRI is capable of analyzing the local environment and density of water protons, and encoding the differences in water properties in the form of contrast in image signal intensity. As tomato develops from green to red, changes in volume element intensity (voxel) and structural features in the images were observed. In the analysis, the voxels in the region of interest (ROI) corresponding to the pericarp tissue of tomatoes were used to calculate

48 statistical features. The partial least square discriminant analysis (PLS-DA) was used to model MR images. PLS-DA modeling provided an effective method of predicting the tomato maturity with an accuracy of 90% for green, breaker-light red and red maturity stages.

In 2005, Karaboga conceptualized the artificial bee colony (ABC) algorithm by modeling the intelligent foraging behavior of honey bees. Bees are grouped into three as employed, onlooker, and scout bees. Employed bees search for profitable food sources and share the information to the onlooker bees. Onlooker bees choose their food sources through probabilistic selection. Some employed bees who abandoned their food sources become scout bees and start looking for new food sources. The optimal food source is found at the end of N cycles. ABC can be used for solving uni-modal and multi-modal numerical optimization problems [28].

Artificial bee colony algorithm is tested by Shah et. al [17] in training multi-layer perceptron (MLP) on earthquake time series data prediction. In this research, real-time series data of seismic event was used in training and testing. During earthquakes, four parameters are being observed namely: depth of earthquake, time of occurrence, geographical area, and the magnitude. The magnitude calculated by the Richter scale was used to train MLP using ABC algorithm. The proper weights of the algorithm may improve the prediction accuracy of the trained MLP. Thus, the simulation results show that ABC can successfully train real-time data for prediction. When compared to back-propagation, ABC shows a significantly higher accuracy.

In the work of Karaboga and Ozturk [29], ABC is tested to perform a clustering type which is fuzzy clustering. In a typical clustering, data points strictly belong to one cluster only. While in fuzzy clustering, data points can belong to more than one cluster and membership degrees between zero and one are used instead of crisp assignments of the data to clusters. The degree of membership in the fuzzy clusters depends on the nearness of the data object on the center of the clusters. In the study, ABC is compared to Fuzzy C-Means (FCM) which is the most popular fuzzy clustering algorithm. Opposite to ABC, FCM easily falls into the local optimal solution because of the random selection of cluster center points. ABC was applied to a collection of benchmark data sets including

Cancer, Diabetes and Heart. Results show that ABC algorithm is very successful on optimization of fuzzy clustering.

A novel data mining approach based on Artificial Bee Colony searching algorithm for classification tasks was introduced by Shukran et. al [9]. As digital information grows rapidly, larger number of raw data needs to be extracted. Data Mining (DM) is the most common method to customize and manipulate data according to what is needed. DM is capable of extracting useful knowledge from large volumes of raw data but this knowledge has to be accurate, comprehensible, readable and easy to understand. Many methods can be used in DM including Evolutionary Algorithms (EA) and other industry standard algorithms. This study used the ABC algorithm in DM which leads to a successful result. Moreover, ABC was compared against six other classifiers such as PART, SOM, Naïve Bayes, Classification Tree and Nearest Neighbor (kNN), and the results reflect the superiority of ABC over the other algorithms.

Zhang et. al [41] confirmed that an improved ABC Algorithm can be very successful in image classification. In their study, the focus is on magnetic resonance (MR) images specifically brain images. Magnetic resonance imaging (MRI) is a noninvasive medical imaging that used radiology to visualize detailed internal structure and limited functions of the body. In medical practice, MR brain images are classified either as normal or abnormal. They proposed a feed-forward neural network (FNN) which is based on scaled chaotic artificial bee colony (SCABC) to find the optimal parameters of the FNN. SCABC is improved by using a fitness scaling strategy and employing a chaotic operator to take place for a number random generator. The novel SCANC-FNN classifier was compared to the traditional FNN based on back-propagation (BP) and genetic algorithm (GA). SCANC-FNN obtained 100% classification accuracy on the T2-weighted brain MRI images datasets.

Dhahri et. al [1] presented an interesting type of neural network which is the beta basis function neural network (BBFNN) used for prediction of benchmark problems. A BBFNN is a three-layer feed-forward network that uses the beta function (a non-linear transfer function) for the hidden units and a linear transfer function for the output units. In effect, the hidden layer is referred to as the beta layer. In this work, BBFNN is

optimized using a new population meta-heuristic – the artificial bee colony (ABC) algorithm. Empirical results show that ABC-BBFNN has an impressive generalization ability giving the smallest training and testing error when compared to other methods.

Chapter 3

Statement of the Problem

Tomatoes are one of the main horticultural crops all over the world. Commercial farms deal with large quantity of tomatoes during harvest time which will be classified and sorted according to maturity or ripeness. Traditionally, this is done by human graders.

Manual tomato grading is prone to error due to various uncontrolled factors. Also, it can be time consuming considering the quantity of tomatoes being produced. Thus, automating this job using image classification techniques will probably help reduce the rate of error and save time.

Artificial neural networks (ANNs) have proven their competitiveness in dealing with classification problems. They have the ability to generalize a new pattern after learning a set of training patterns. But the success of the neural network depends on many factors such as the network structure, training features, and the training algorithm used.

Since an artificial neural network is focused on weights optimization, various optimization algorithms have been employed to do the task. Swarm intelligence (SI) has the ability not to get stuck in local optima making it a more efficient trainer of an ANN compared to traditional methods. Artificial Bee Colony (ABC), a relatively new swarm-based algorithm, has emerged which was motivated by the intelligent foraging behavior of honey bees.

In this study, ABC will be tested to train a feed-forward neural network to classify tomato images.

Chapter 4

Objectives

The objectives of this study are as follows:

1. To test the effectiveness of ABC-NN Classifier in tomato maturity classification.
2. To determine the effect of varying the parameter values of the ABC-NN Classifier.
3. To determine the effect of combining the components of three color models (RGB, HSI, CIE L*a*b*) as inputs to the classifier

Chapter 5

Proposed Approach

5.1 General Flow

Figure 5.1 shows the general flow diagram of the proposed approach.

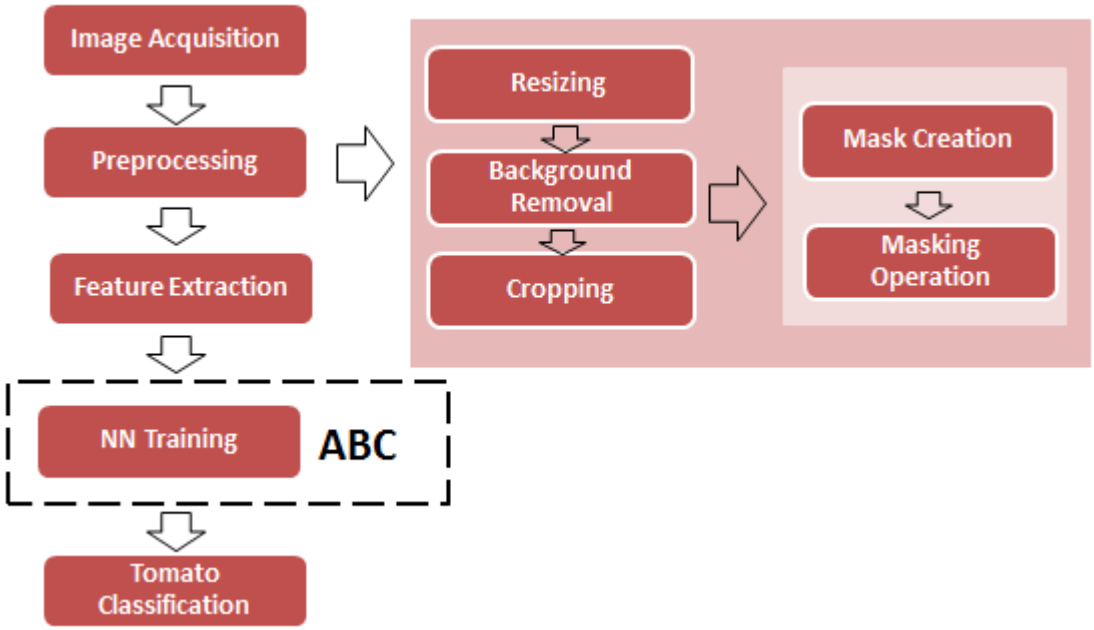


Figure 5.1: Program flow diagram

5.2 Input

The inputs for this study are M x N color images of tomatoes. Five color features will be extracted from each image which will then be represented as an input vector to the neural network classifier. Three of the five features are from the RGB color space which includes Red (R), Green (G), and Red minus Green (R-G) components. The other two features are Hue (H) from HSI and a* from CIELAB. These five features have been the focus of the past literatures in relation to tomato maturity classification. Figure 5.2 shows sample input images.

Figure 5.2: Sample input images

5.3 Process

5.3.1 Image Acquisition

Tomato images are acquired using a Sony digital camera. To simplify the background removal step, a plain white paper is used as the background for capturing tomato images as shown in Figure 5.2. But the images acquired are quite large and contain the background area. Hence, there is a need to preprocess these images before proceeding to the feature extraction phase.

5.3.2 Image Preprocessing

Preprocessing tomato images is important so that the values of the extracted color features are correct. Tomato preprocessing involves resizing, background removal, and cropping as shown in Figure 5.1.

5.3.2.1 Image Resizing

Tomato images undergo image resizing twice in the entire process. The first one is done before background removal wherein images are to be resized down to 200 x 200 pixels. This is to make the background removal process much faster. The second one is done

after image cropping when tomato images have different sizes. After the second resizing, tomato images will have a size of 64 x 64 pixels and are now ready for feature extraction.

5.3.2.2 Background Removal

To remove the background, the following steps will be applied to each image:

1. Convert to blue channel
2. Convert to grayscale
3. Create binary mask
4. Remove background by applying the binary mask

The very first step in the background removal phase is the conversion of an RGB image into a blue channel image. Since the blue channel is just a subset of the RGB model, we can easily accomplish this step by just setting the values of red and green elements to zero. In effect, only the blue element will retain its value as shown in Figure 5.3a.

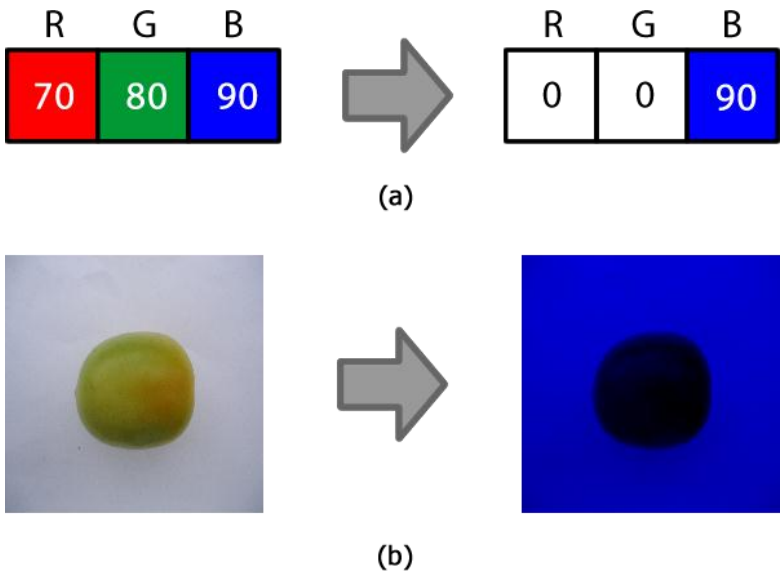


Figure 5.3: RGB to blue channel conversion

After converting to blue channel, the image will be converted to grayscale. Among the three formulas for grayscale conversion discussed in the introduction, the averaging formula shown in equation (9) will be used. As seen in Figure 5.4a, a grayscale

pixel contains identical values for the three color elements. In effect, a grayscale pixel can be represented by a single value.

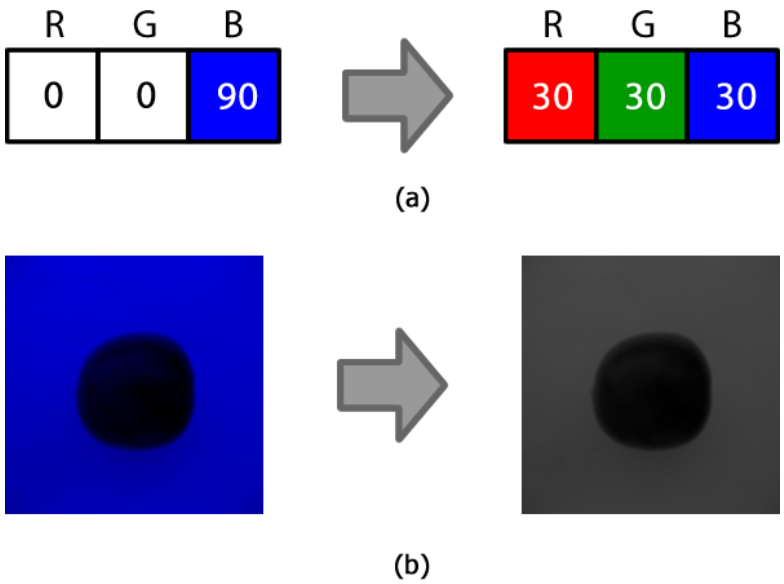


Figure 5.4: Grayscale conversion

After grayscale conversion, we are now ready to create a binary mask using the Otsu thresholding technique. The first step is to create a histogram of the grayscale image as shown in Figure 5.5. The histogram is actually represented by an array of integers with a size of 256 as shown in Figure 5.6. But as we can see in Figure 5.5, the histogram is only extended up to 90 units because the maximum gray value of the image is only 72.

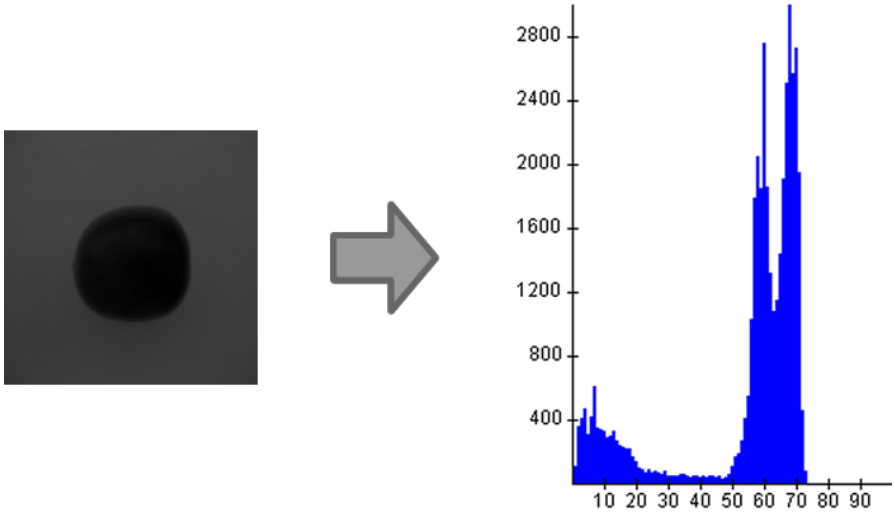


Figure 5.5: Histogram of the grayscale

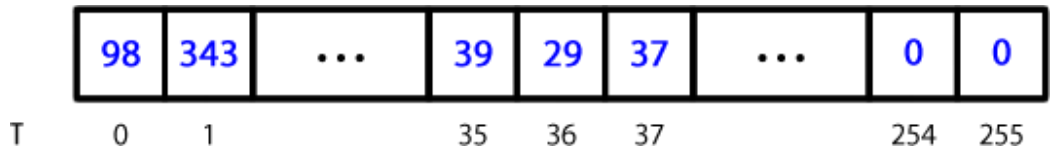


Figure 5.6: Array representation of the histogram in Figure 5.7

Once we have the histogram array, we can now find the optimal threshold value *T* using Otsu's method. The goal is to find a threshold that maximizes the between-class variance with the use of equation (27). Otsu's method will iterate from index 0 through 255 to find the optimal threshold value. At index = 36, the between-class variance is at maximum. The calculations are as follows:

$$\boldsymbol{\sigma^2_{Between}(T)} = \omega_0(T)\omega_1(T)[\mu_0(T) - \mu_1(T)]^2 \qquad \textbf{(27)}$$

Probability and mean of Class 0:

$$\omega_0(36) = \sum_{i=0}^{36} p(i)$$

$$\omega_0(36) = 98 + 343 + \cdots + 39 + 29$$

$$= 6,696$$

$$\mu_0(36) = \frac{\sum_{i=0}^{36} p(i)*i}{\omega_0(36)}$$

$$\mu_0(36) = [(0 * 98) + (1 * 343) + \cdots + (35 * 39) + (36 * 29)]/6696$$

$$\approx 10.71$$

Probability and mean of Class 1:

$$\omega_1(36) = \sum_{i=37}^{255} p(i)$$

$$\omega_1(36) = 37 + \cdots + 0 + 0$$

$$= 33,304$$

$$\mu_1(36) = \frac{\sum_{i=37}^{255} p(i)*i}{\omega_1(36)}$$

$$\mu_0(36) = [(37 * 37) + \cdots + (254 * 0) + (255 * 0)]/\ 33304$$

$$\approx 62.54$$

From equation (27):

$$\sigma^2_{Between}(36) = 6,696 * 33,304 * (10.71 - 62.54)^2$$

$$\approx 5.990E11$$

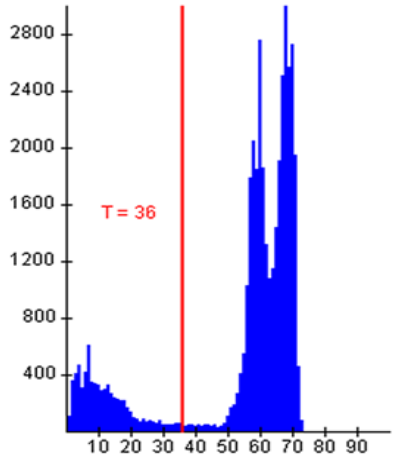


Figure 5.7: Histogram, *T* = 36

Since at index = 36 the between-class variance is at maximum, the optimal threshold becomes 36. The threshold will determine which pixels belong to foreground and which pixels belong to the background. Pixels at the left of the threshold are considered the foreground pixels (represented by white color) while those at the right of the threshold are considered the background pixels (represented by black color). Shown in Figure 5.8 is the binary mask generated after applying image binarization using T = 36.

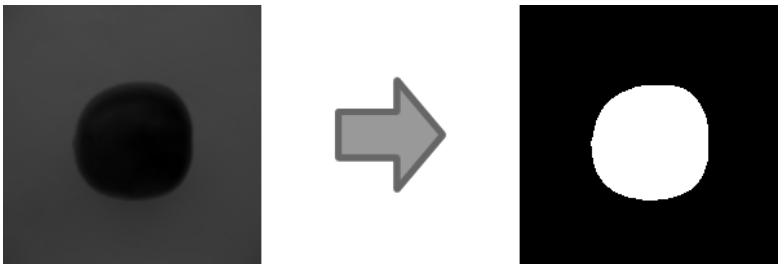


Figure 5.8: Binary mask creation

The final step in background removal is the masking operation or applying the binary mask into the image. By multiplying the original image and the mask, we can extract the region of interest from the original image as shown in Figure 5.9c. If the pixel

value of the mask is 1, the product will just be the same as the original pixel (Figure 5.9a). However if the pixel value of the mask is 0, the product will be the same as the mask pixel (Figure 5.9b).

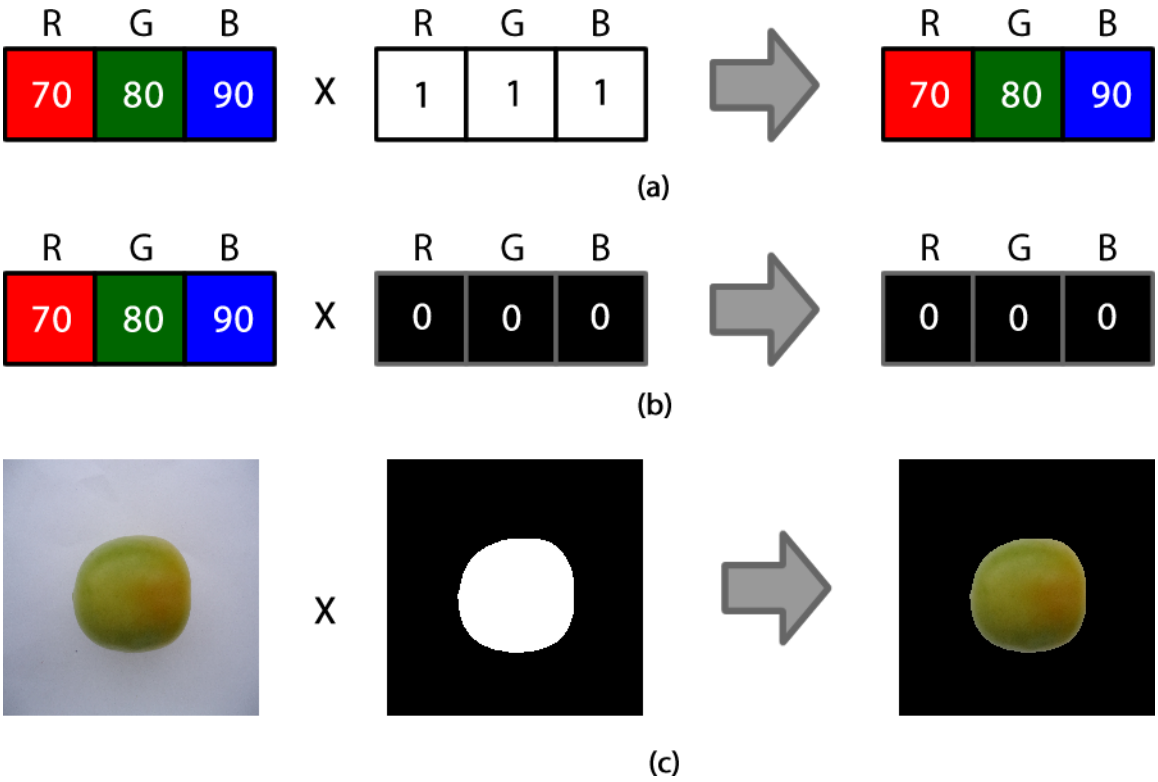


Figure 5.9: Masking operation

5.3.2.3 Image Cropping

Background pixels have no use in the feature extraction phase. Having more of these pixels is inefficient because it will contribute to the increase in processing time. To reduce useless pixels, each tomato image will undergo image cropping. In Figure 5.10a, useless pixels are represented by black color. This will make the cropping process easier by just getting the lower and upper bounds of the foreground pixels. Figure 5.10b shows the cropped image with lesser number of useless pixels.

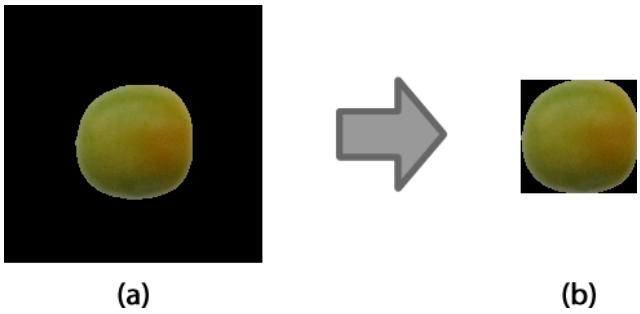


Figure 5.10: Image cropping

5.3.3 Feature Extraction

One of the factors that can affect the success of a neural network is the selection of the features to be used in the training and testing process. In this paper, color features will be used to determine the tomato maturity.

Unlike other literatures, this paper will be utilizing three color models namely: RGB, HSI and L*a*b*. Five color features will be extracted including average red (RGB), average green (RGB), average of red-green difference (RGB), average hue (HSI), and average a* (L*a*b*). Extracting the RGB features will be easy since the input image is in RGB format. However, to be able to extract the hue and a* features there is a need to transform the RGB format into HSI and L*a*b* using equations (1, 2, 3) and (5, 6, 7), respectively. Figure 5.12 shows the corresponding color features of the sample input image.

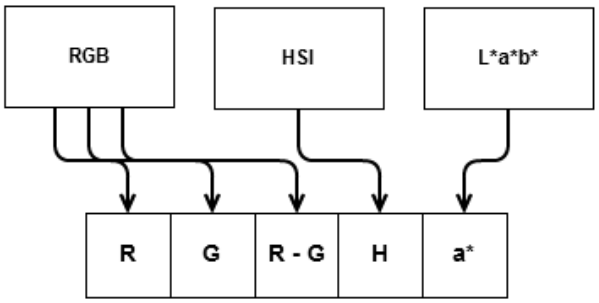


Figure 5.11: Color features origin

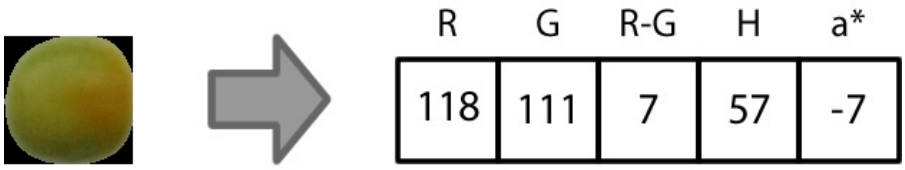


Figure 5.12: Feature extraction

5.3.4 Neural Networks Training By ABC

The ABC algorithm is responsible for finding the optimal solution (food source) by searching through the solution space. ABC will train the neural network using the following parameters:

Number of employed bees (BN) – corresponds to the initial number of solutions

Number of onlooker bees (NW) – corresponds to the maximum number of onlooker bees that will exploit a particular solution

Number of scout bees – 10% of the number of employed bees

Maximum cycle number (MCN) – maximum number of iterations of the ABC algorithm

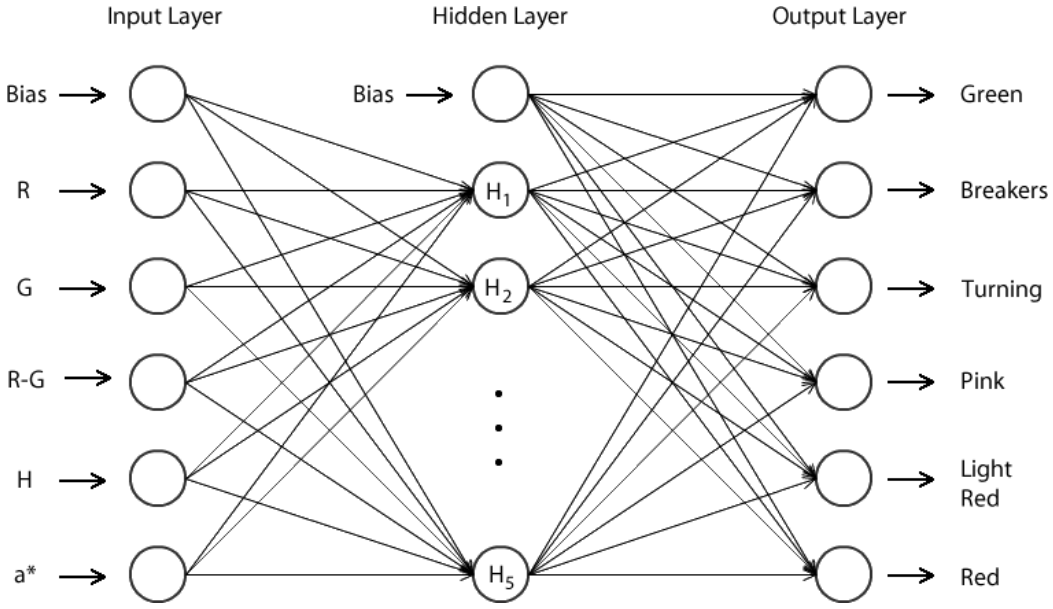


Figure 5.13: The neural network structure

5.3.4.1 Population Initialization

A food source X will be defined as a weight vector associated to a neural network. A food source represents a solution to the problem.

$$X_i = x_1 x_2 x_3 x_4 \dots x_D \quad \textbf{(28)}$$

where x is the weight of a connection from a node to another node and D is the dimension of the neural network. The food sources will be initialized randomly with a size depending on the number of employed bees. Each parameter j of food source i will be randomly generated using equation (29).

$$X_{ij} = l_j + r * (u_j - l_j) \quad \textbf{(29)}$$

where u_j is the upper bound equal to 1; l_j is the lower bound equal to -1; and r is a random number from [0, 1]. In the sample generated solution in Figure 5.14a, the first parameter (j = 1) got a random value r = 0.44860973448190555 that resulted to the following computation:

x_{ij} = -1 + 0.44860973448190555 * [1 – (-1)]

= **-0.1027805310361889**

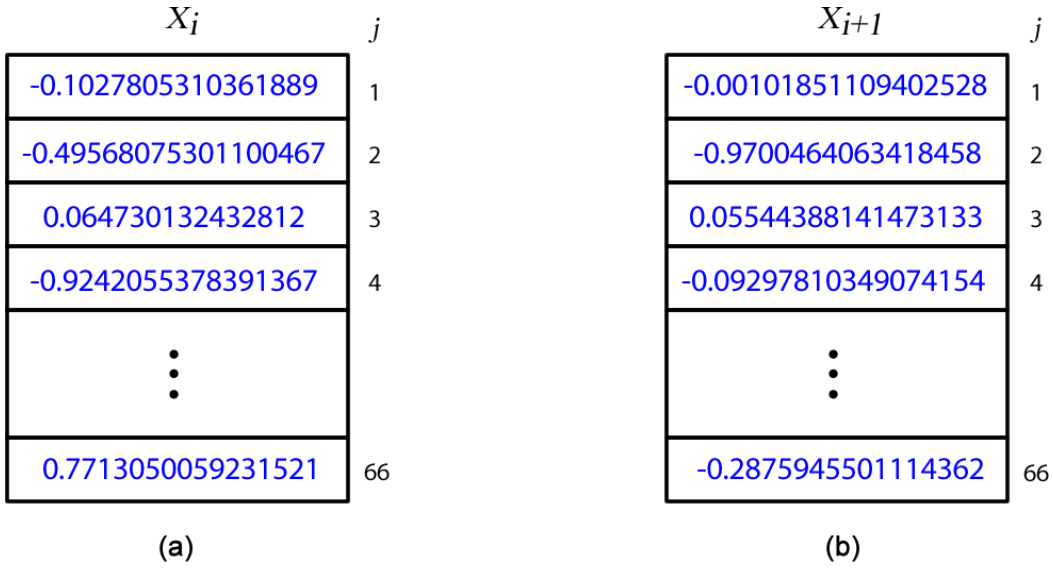


Figure 5.14: Sample generated food sources

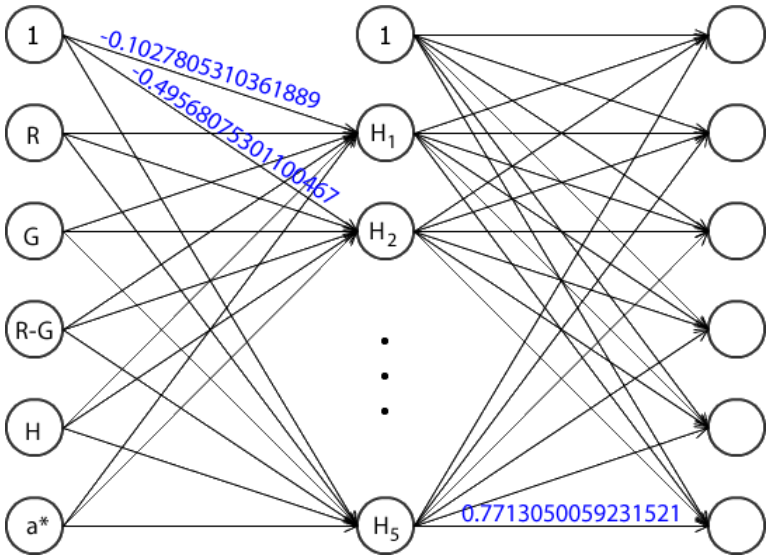


Figure 5.15: Neural network equivalent of the food source in Figure 5.14a

5.3.4.2 Neighborhood Search

In neighborhood search, the goal is to try to improve the current solution by looking for a neighboring solution. Both the employed and onlooker bees perform neighborhood search using equation (30).

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{hj}) \quad \textbf{(30)}$$

When i = 1, and BN = 15, h can be randomly selected through 2 to 15. Then the parameter index j is randomly selected among the D-dimensional parameters from 1 to 66. Once we have these values we can now compute for the parameter j of the neighboring solution of i. To illustrate the neighborhood search,

Let i = solution in Figure 5.14a

h = solution in Figure 5.14b

j = 1

$\emptyset$ = 0.45457118383043693

Thus,

V_{ij} = -0.1027805310361889 + 0.45457118383043693 *
(-0.1027805310361889 – (-0.00101851109402528))

= **-0.14903861291027475**

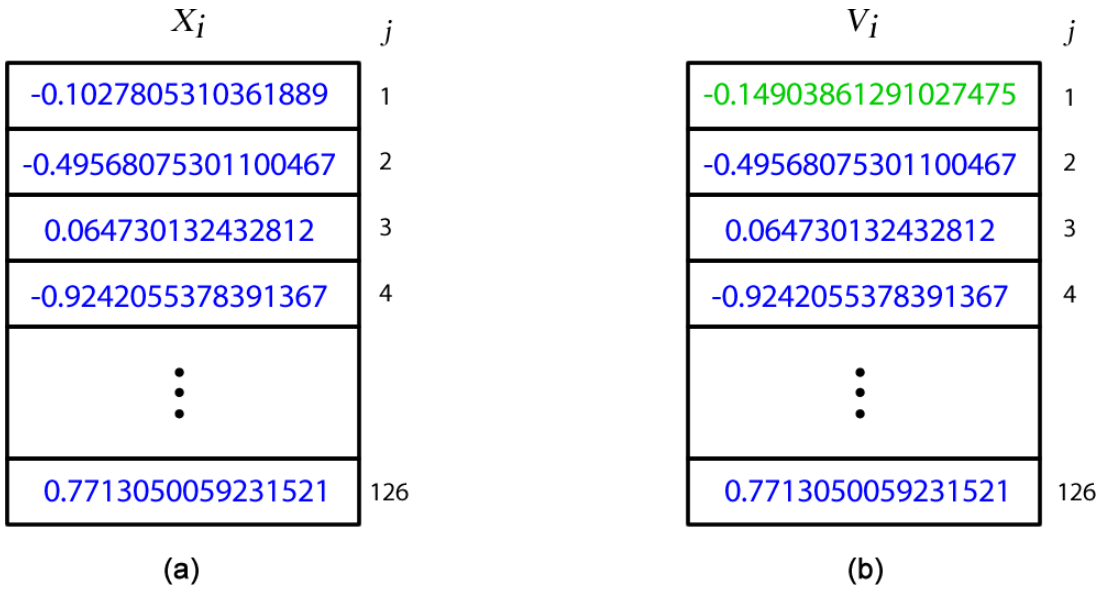


Figure 5.16: (a) Original solution X_i, (b) New solution V_i where j = 1

5.3.4.3 Population Evaluation

Since the solutions are represented by vector weights of the neural networks, the mean square error (MSE) as shown in equation (31) will be used to assess the accuracy of the solution.

$$MSE = \frac{1}{N*M}\sum_{i=1}^{N}\sum_{j=1}^{M}(T_{ij} - A_{ij})^2 \quad \textbf{(31)}$$

where N = Number of training patterns

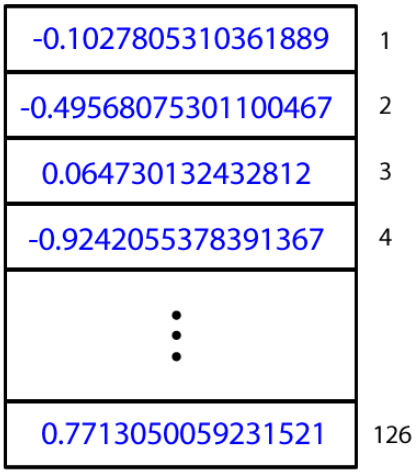
M = Number of output nodes

T_{ij} = Target value of pattern i node j

A_{ij} = Actual value of pattern i node j

$$fit_i = \frac{1}{1+f_i} \quad \textbf{(32)}$$

Given a solution in Figure 5.16a, the mean square error will be calculated using equation (31). Then using the fitness function in equation (32), we can now determine the quality or nectar amount of the solution by substituting f_i with the *MSE*.

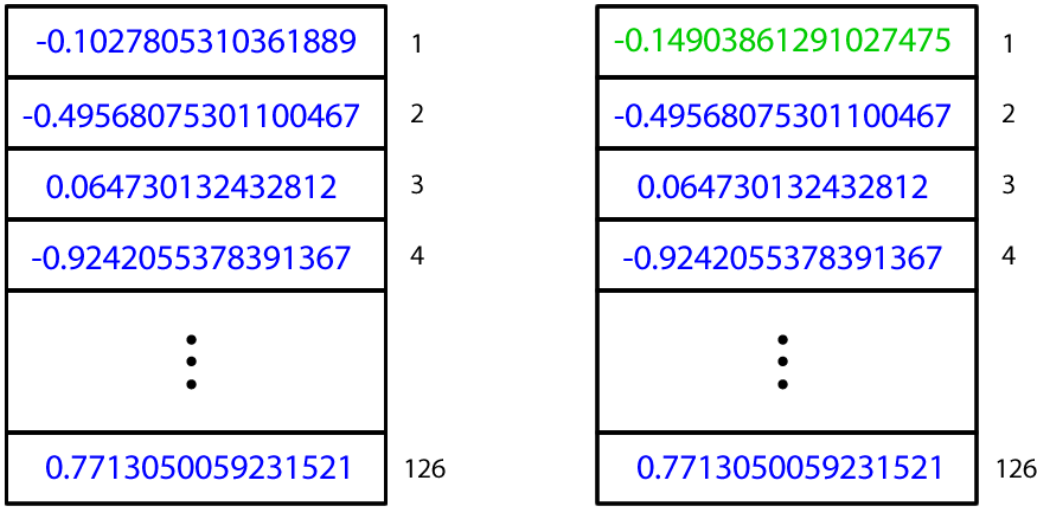


$MSE_i = 0.2124769453315525$ $fit_i = 0.8247579501204861$

Figure 5.17: Solution X_i and its corresponding MSE and fitness value

5.3.4.4 Greedy Selection

Right after the neighborhood search, a greedy selection will be performed between the current solution X_i and the newly generated solution V_i. If the fitness value of V_i is equal or higher than X_i, V_i will replace X_i in the memory. Otherwise, the solution X_i is retained in the memory.



$fit(X_i) = 0.8247579501204861$ $fit(V_i) = 0.8265062811402341$

Figure 5.18: Solutions X_i and V_i with their corresponding fitness values

5.3.4.5 Onlooker Bee Phase

During the onlooker bee phase, the probability of a solution to be exploited by the onlooker bees depends on the probability function given in equation (33):

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad \textbf{(33)}$$

The probability is proportional to the nectar amount of the food source. The higher the fitness value of a solution means that the higher will be its probability to be chosen by onlooker bees. Once a food source is chosen, the number of onlooker bees to be assigned to that specific food source is determined by equation (34).

$$N_i = P_i * NW \quad \textbf{(34)}$$

For example, consider the solution V_i in Figure 5.18 which has a fitness value of 0.8265062811402341. Let us say that SN = 15 and $\sum_{n=1}^{15} fit_n$= 4.0024322895, then P_i = 0.2065010027. If NW = 15, then N_i = 3.0975150405 and by rounding off becomes 3. This means that solution V_i will operate through onlooker bee phase 3 times.

5.3.4.6 Scout Bee Phase

Scouting is the process of possibly replacing worst solutions, solutions with the lowest fitness values, with randomly generated new solutions. For each worst solution, a random solution is generated and then a greedy selection is applied between the two. The number of worst solutions is equal to the number of scout bees which is 10% of the total number of employed bees. Replacement is an important concept in ABC because it allows solutions of poor quality to be replaced by possibly better solutions. In Figure 5.19, the new solution will replace the current solution because it has a higher fitness value.

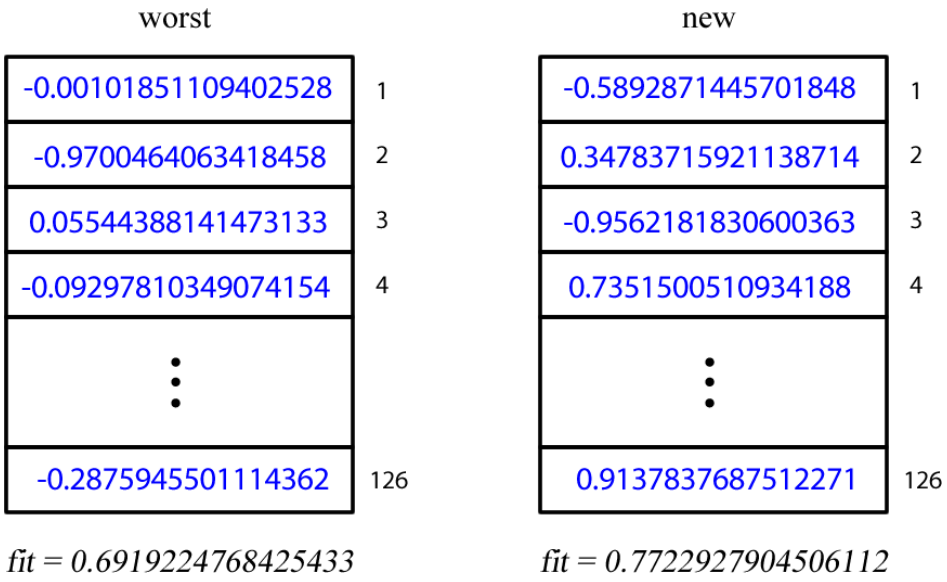


Figure 5.19: Worst solution and newly generated solution

5.4. Output

After a certain number of cycles, an optimal solution with the least mean square error (MSE) is produced as a result of the training phase. Shown in Figure 5.20 is a sample optimal solution and its corresponding neural network is shown in Figure 5.21 which will be used as a classifier during the classification phase.

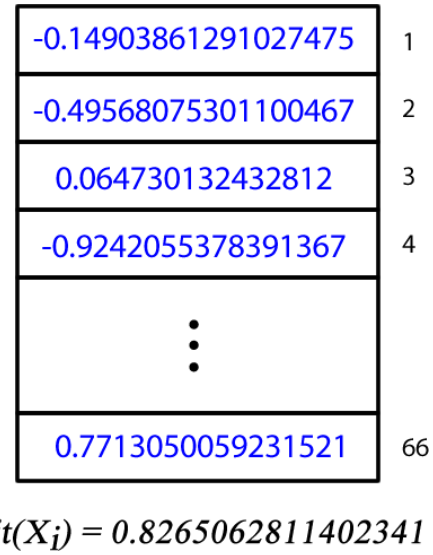


Figure 5.20: Sample optimal solution

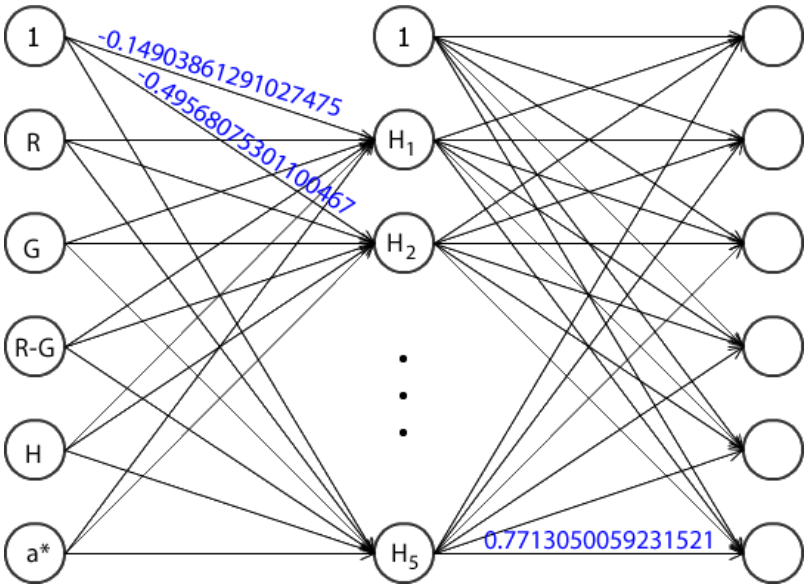


Figure 5.21: Optimal neural network

In the classification phase, the features extracted from a tomato image will be used as inputs to the neural network classifier. In Figure 5.22, if we feed the color features from Figure 5.12 to the classifier then the resulting output vector is [0, 1, 0, 0, 0, 0] which means that the tomato is in the Breakers stage. The actual values of the output nodes are in decimal formats of the range [0, 1] but we set the node with the highest value to 1 while the others to 0.

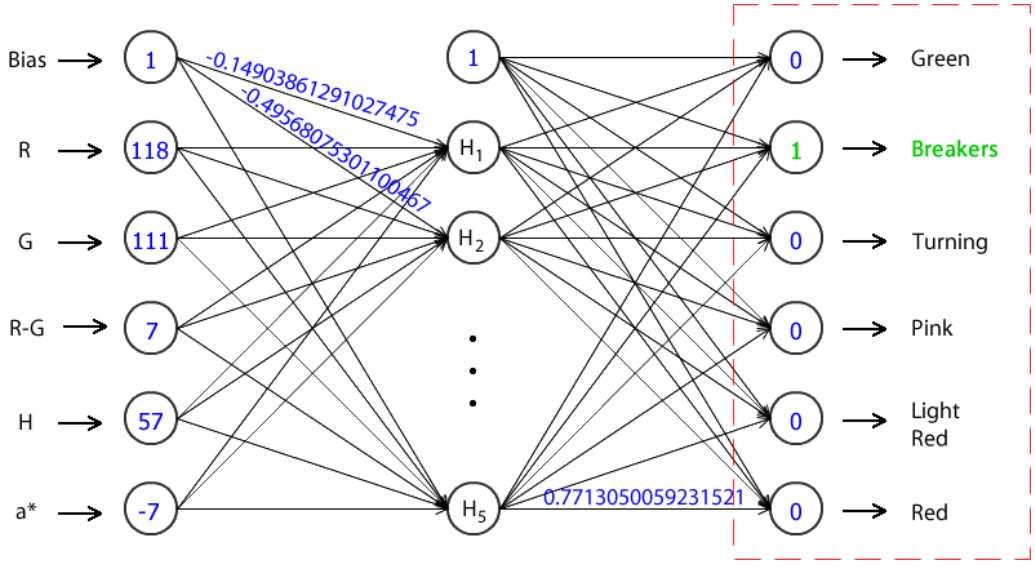


Figure 5.22: A sample output of the neural network classifier

Chapter 6

Experiments and Results

6.1 Experimental Setup

The system used for experimentation has the following specifications:

Hardware

Intel(R) Core(TM) i3-2310M CPU @ 2.10 GHz

2.0 GB of RAM

Software

Microsoft Windows 7 Ultimate 32-bit (6.1, Build 7600)

Java JDK 1.7.0_04 though Eclipse IDE 4.2.0

6.2 Results and Discussions

The program was coded in Java using Eclipse IDE. The dataset is composed of six hundred (600) images of tomatoes which were used in the experiments. These images were acquired using a digital camera and were manually classified according to their maturity stages. The effectiveness and efficiency of the ABC-NN classifier will be tested based on the manual classification of tomatoes. To evaluate the performance of the ABC-NN classifier in tomato maturity classification, the dataset was randomly divided into two sets. Seventy percent (70%) of it was used in training and the remaining thirty percent (30%) was used in testing the classifier.

To assess the effectiveness of the ABC-NN classifier, we define the standard parameter values of the ABC algorithm and neural networks as shown in Table 6.1 and Table 6.2, respectively. The program was run 30 times in all experimental setups to ensure the statistical acceptability of the results. Also, we use both the training set and testing in the classification to see how the classifier performs.

Table 6.1: ABC standard parameter values

Parameters	Standard Values
Number of Employed Bees (BN)	15
Number of Onlooker Bees (NW)	15
Max Cycle (MCN)	750

Table 6.2: NN standard structure

Parameters	Standard Values
Numb of Hidden Layer	1
Number of Hidden Nodes	5 (not including the bias node)
Dimension	66

For the first experimentation, two types of dataset were tested. The fixed setup uses a fixed training set and a fixed test set in the entire 30 runs. On the other hand, the training set and test set were re-shuffled in each run of the random setup.

Table 6.3: Results of ABC-NN Classifier in 30 runs

	Fixed		Random	
	Test Set	**Training Set**	**Test Set**	**Training Set**
Mean	98.19 %	98.73 %	97.81 %	98.75 %
Median	98.34 %	98.81 %	97.50 %	98.81 %
Maximum	100.0 %	99.52 %	99.44 %	99.52 %
Minimum	96.11 %	96.90 %	95.56 %	97.14 %
SD	1.041224	0.620035	1.081337	0.518193
Variance	1.084147	0.384443	1.169289	0.268524

Table 6.3 shows the results of the test for effectiveness of the ABC-NN classifier. It can be observed that the classifier performed well in both the fixed dataset and random dataset in both the test set and training set experiment. In terms of precision, the results show small values of standard deviation and variance.

If we analyze the results, the training set got higher results than the test set in all aspects except for the maximum value in the fixed setup. It is normal for the training set to have better results than test set because it was the one used in modeling the neural network. But here, the maximum accuracy for the training set is only 99.52% or 418 out of 420 samples are correctly classified. This means that there are two image samples that are misclassified which are part of the training set.

The next experiment conducted was the variation of parameters for both the ABC algorithm and neural networks. These include varying the number employed bees, onlooker bees, MCN, and hidden nodes. For the following experiments, we use the fixed dataset for the whole 30 runs. We define the parameter variations in the following tables.

Table 6.4: Varying the Number of Employed Bees

Set	Employed Bees	Onlooker Bees	Max Cycle
A	**5**	15	750
B	**10**	15	750
C	**15**	15	750
D	**20**	15	750
E	**25**	15	750

Table 6.5: Varying the Number of Onlooker Bees

Set	Employed Bees	Onlooker Bees	Max Cycle
A	15	**5**	750
B	15	**10**	750
C	15	**15**	750
D	15	**20**	750
E	15	**25**	750

Table 6.6: Varying the Number of Maximum Cycles

Set	Employed Bees	Onlooker Bees	Max Cycle
A	15	15	**250**
B	15	15	**500**
C	15	15	**750**
D	15	15	**1000**
E	15	15	**1250**

Table 6.7: Varying the Number of Hidden Nodes

Set	Number of Hidden Nodes
A	**2**
B	**3**
C	**5**
D	**10**
E	**15**

Table 6.8 and Figure 6.1 show the time needed for the different setups to train the neural networks. The training time is proportional to the number of employed bees. Since the number of employed bees determines the number of solutions and each solution is to be optimized, increasing the number of employed bees means that more time is needed in the optimization process.

Table 6.8: Training time results (in seconds) when varying the number of employed bees

	A	B	C	D	E
Mean	49.88	100.03	153.46	200.33	253.98
Median	49.96	100.26	151.77	200.40	253.36
Maximum	52.00	104.97	173.32	204.76	264.93
Minimum	46.86	95.61	146.10	193.49	246.39
SD	1.221656	1.891891	5.873779	2.511303	4.540168
Variance	1.492442	3.579252	34.50128	6.306643	20.61312

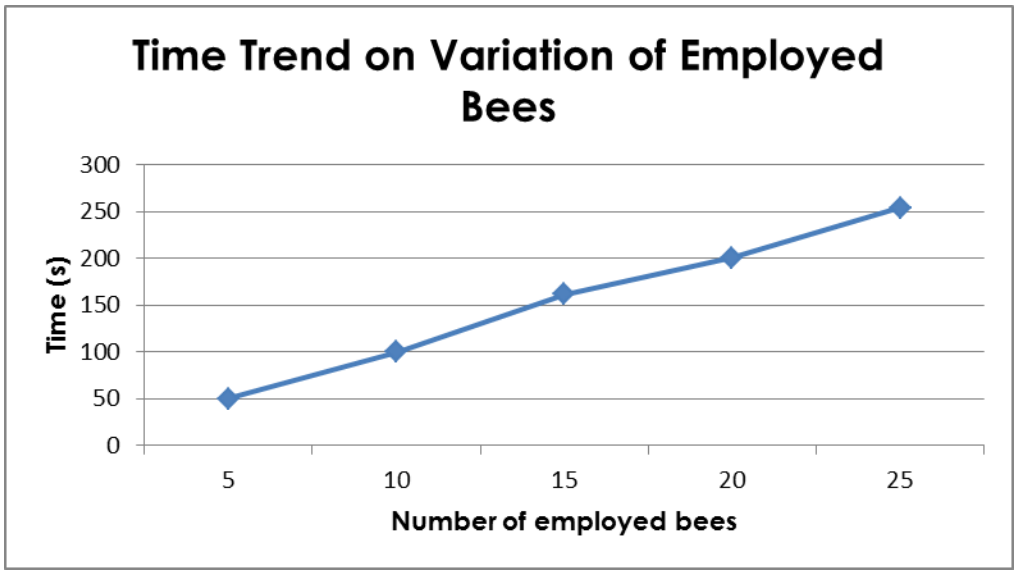


Figure 6.1: Average training time on varying the number of employed bees

In Table 6.9 and Table 6.10, we can see the effect of varying the number of employed bees in terms of the accuracy rate, as well as in terms of the standard deviation and variance. When the number of employed bees is only 5, we can see poor results such as the minimum accuracy equals to 66.11% in the testing set and 67.62% in the training set. Also, there is a very high standard deviation and variance for setup A compared to the other setups.

Since the number of employed bees equals the number of solutions, increasing this value increases the solution space. Thus, it can be assumed that it will affect the optimality of the solution or the classification accuracy. The results support this assumption as the accuracy increases along with the increase in the number of employed bees. It can be clearly seen in Figure 6.2 that having a too small solution space results to poor accuracy rate.

Table 6.9: Test set results when varying the number of employed bees

	A	B	C	D	E
Mean	93.28 %	97.78 %	98.19 %	98.26 %	98.28 %
Median	96.11 %	98.06 %	98.34 %	98.34 %	98.34 %
Maximum	98.33 %	98.89 %	100.0 %	100.0 %	100.0 %
Minimum	66.11 %	96.11 %	96.11 %	96.67 %	97.22 %
SD	7.227893	0.922727	1.041224	0.884246	0.704995
Variance	52.24244	0.851425	1.084147	0.781891	0.497018

Table 6.10: Training set results when varying the number of employed bees

	A	B	C	D	E
Mean	94.11 %	98.41 %	98.73 %	98.74 %	98.84 %
Median	97.14 %	98.69 %	98.81 %	98.81 %	99.05 %
Maximum	98.81 %	99.29 %	99.52 %	99.52 %	99.29 %
Minimum	67.62 %	95.48 %	96.90 %	97.14 %	98.10 %
SD	7.257679	0.834993	0.620035	0.492931	0.399061
Variance	52.67391	0.697213	0.384443	0.242981	0.159249

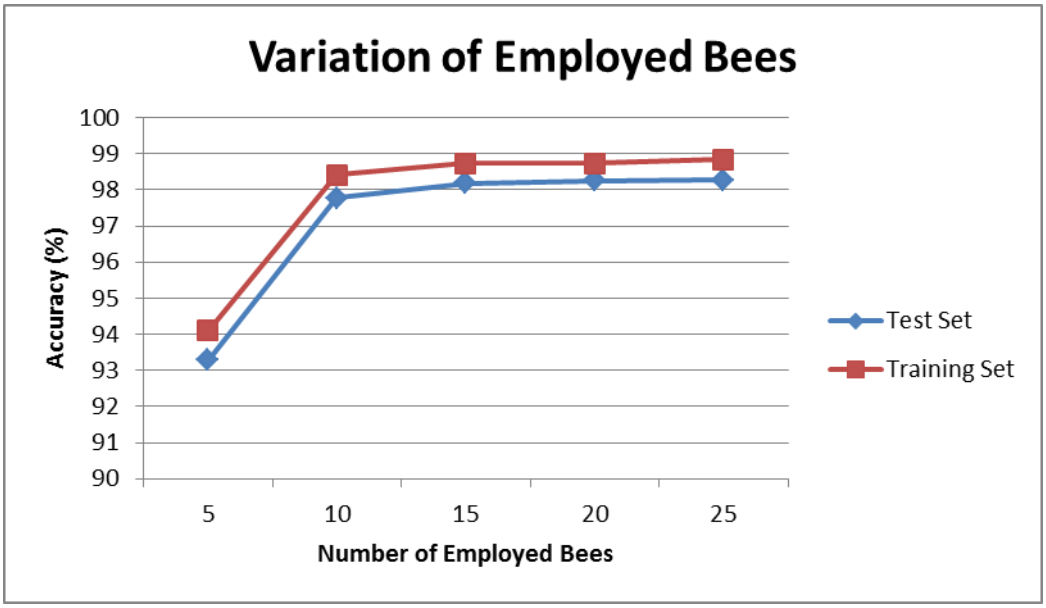


Figure 6.2: Mean accuracy when varying the number of employed bees

Table 6.11 shows the time results in neural networks training when varying the number of onlooker bees. The results are quite the same with the employed bee's experiment in which the training time is proportional with the number of onlooker bees.

Since the number of onlooker bees determines the number of times a specific food source is to be exploited during the onlooker bee's phase, then increasing this value means that more exploitation and exploration time will be consumed. This trend can be seen in Figure 6.3.

Table 6.11: Training time results (in seconds) when varying the number of onlooker bees

	A	B	C	D	E
Mean	48.54	99.49	153.46	204.20	256.46
Median	48.57	99.47	151.77	204.30	256.74
Maximum	49.65	103.33	173.32	209.75	264.89
Minimum	47.63	96.57	146.10	194.66	247.50
SD	0.534060	1.566220	5.873779	3.485664	4.854389
Variance	0.285221	2.453044	34.50128	12.14985	23.56509

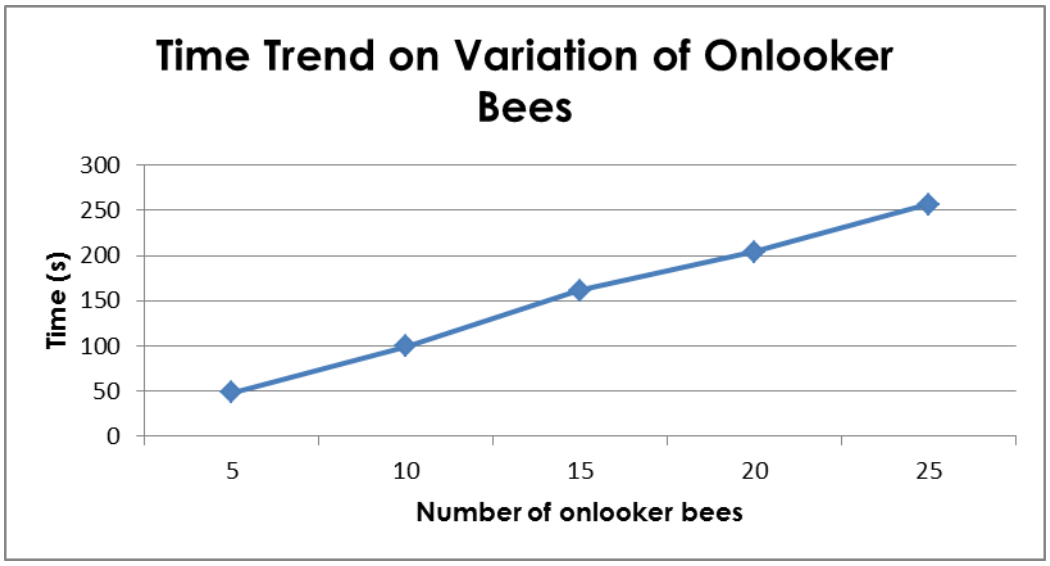


Figure 6.3: Average training time on varying the number of onlooker bees

As mentioned in the previous paragraph, onlooker bees exploit a food source and explore for neighboring food sources to find higher nectar amount. Hence, having more onlooker bees will increase the chance of finding better food sources or solutions.

In Table 6.12, Table 6.13, and Figure 6.4, the results show a slight trend in the accuracy in both the test set and training set. But unlike the employed bee's experiment, having only 5 onlooker bees still resulted to a high accuracy. This is because the number of food sources or the size of the solution space is enough to search for good solutions. Increasing the number of onlooker bees slightly increased the overall results.

Table 6.12: Test set results when varying the number of onlooker bees

	A	**B**	**C**	**D**	**E**
Mean	96.28 %	97.44 %	98.19 %	98.11 %	98.16 %
Median	96.67 %	97.78 %	98.34 %	98.33 %	98.33 %
Maximum	98.89 %	99.44 %	100.0 %	100.0 %	99.44 %
Minimum	87.22 %	95.00 %	96.11 %	96.67 %	97.22 %
SD	2.144970	1.276901	1.041224	0.752462	0.716971
Variance	4.600897	1.630477	1.084147	0.566199	0.514047

Table 6.13: Training set results when varying the number of onlooker bees

	A	**B**	**C**	**D**	**E**
Mean	97.53 %	98.29 %	98.73 %	98.77 %	98.90 %
Median	97.74 %	98.33 %	98.81 %	99.05 %	99.05 %
Maximum	99.29 %	99.33 %	99.52 %	99.52 %	99.52 %
Minimum	93.33 %	96.90 %	96.90 %	96.14 %	97.86 %
SD	1.298819	0.713143	0.620035	0.676571	0.452977
Variance	1.686931	0.508574	0.384443	0.457748	0.205188

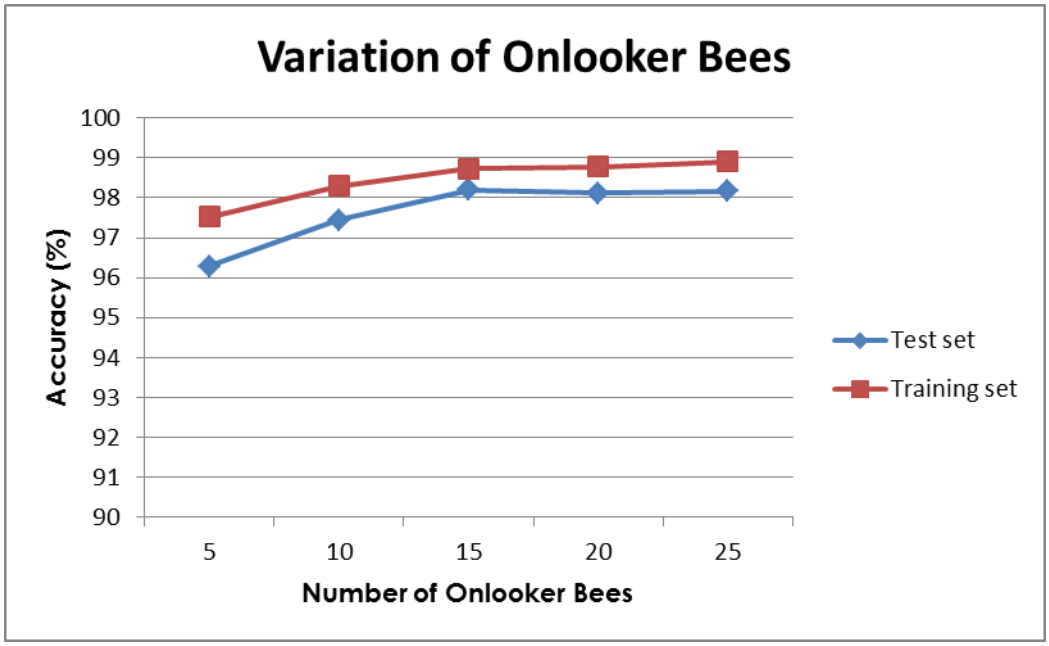


Figure 6.4: Mean accuracy when varying the number of onlooker bees

The time results in the variation of MCN in training the neural networks are shown in Table 6.14. We can observe that for all the variation of ABC parameters experiments, the time trends are the same all throughout. The time needed for training the neural networks increases as the ABC parameter value increases.

The number of cycles determines the number of times the ABC cycle will iterate. Each cycle consumes time for the foraging bees to find for solutions. So, increasing the cycle number increases the ABC processing time.

Table 6.14 Training time results (in seconds) when varying the MCN

	A	B	C	D	E
Mean	47.74	98.93	153.46	205.31	257.41
Median	46.86	98.27	151.77	205.31	256.93
Maximum	63.13	115.35	173.32	214.26	266.59
Minimum	45.66	93.33	146.10	195.90	251.16
SD	3.309218	3.732983	5.873779	3.839589	3.986874
Variance	10.95092	13.93516	34.50128	14.74245	15.89516

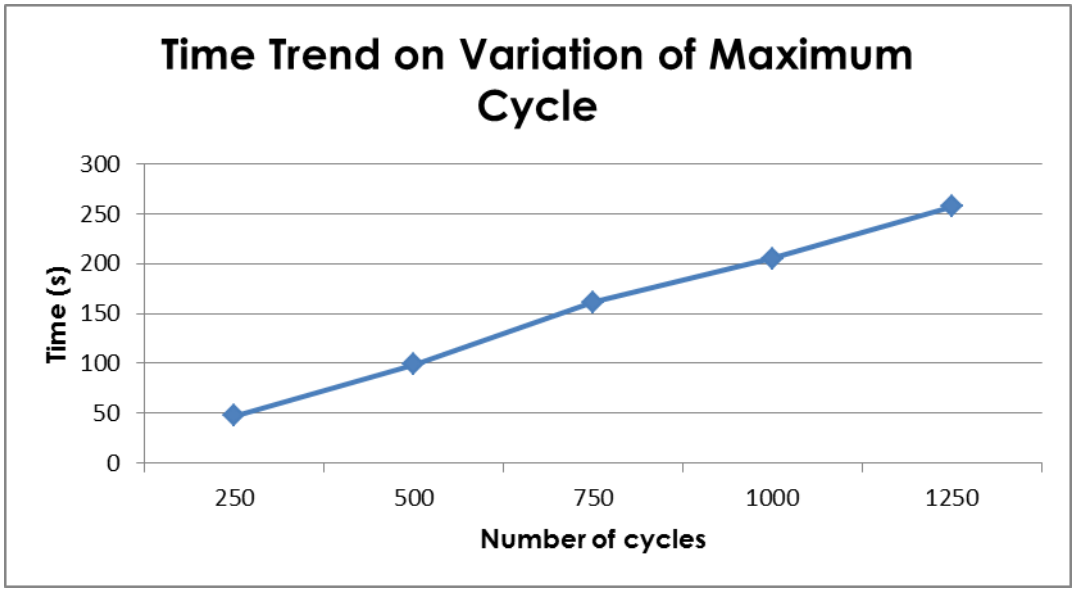


Figure 6.5: Average training time on varying the number of cycles

The classification results for the test set and training set are shown in Table 6.15 and Table 6.16, respectively. Figure 6.6 shows the trend showing that as the MCN increases, there is a slight increase in the accuracy results in both setups.

Just like the other ABC parameters, the MCN parameter allows for the solutions in the solution space to be further improved. Basically if the MCN is set to zero, then the randomly initialized solutions are the basis for choosing the optimal solution. But as the MCN increases there are higher chances of improving the solutions. Based on the experimental results, setups with higher MCN tend to produce better solutions resulting to higher accuracy rate.

Table 6.15: Test set results when varying the MCN

	A	B	C	D	E
Mean	97.02 %	97.44 %	98.19 %	98.12 %	98.19 %
Median	97.222 %	97.78 %	98.34 %	98.33 %	98.33 %
Maximum	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
Minimum	92.78 %	93.89 %	96.11 %	94.44 %	96.11 %
SD	1.685609	1.541259	1.041224	1.125229	0.849878
Variance	2.841279	2.375481	1.084147	1.266142	0.722293

Table 6.16: Training set results when varying the MCN

	A	B	C	D	E
Mean	97.97 %	98.26 %	98.73 %	98.71 %	98.87 %
Median	99.33 %	98.57 %	98.81 %	98.81 %	99.05 %
Maximum	99.29 %	99.29 %	99.52 %	99.29 %	99.52 %
Minimum	93.81 %	95.24 %	96.90 %	97.38 %	97.62 %
SD	1.189245	0.915009	0.620035	0.457771	0.449512
Variance	1.414303	0.837242	0.384443	0.209554	0.202061

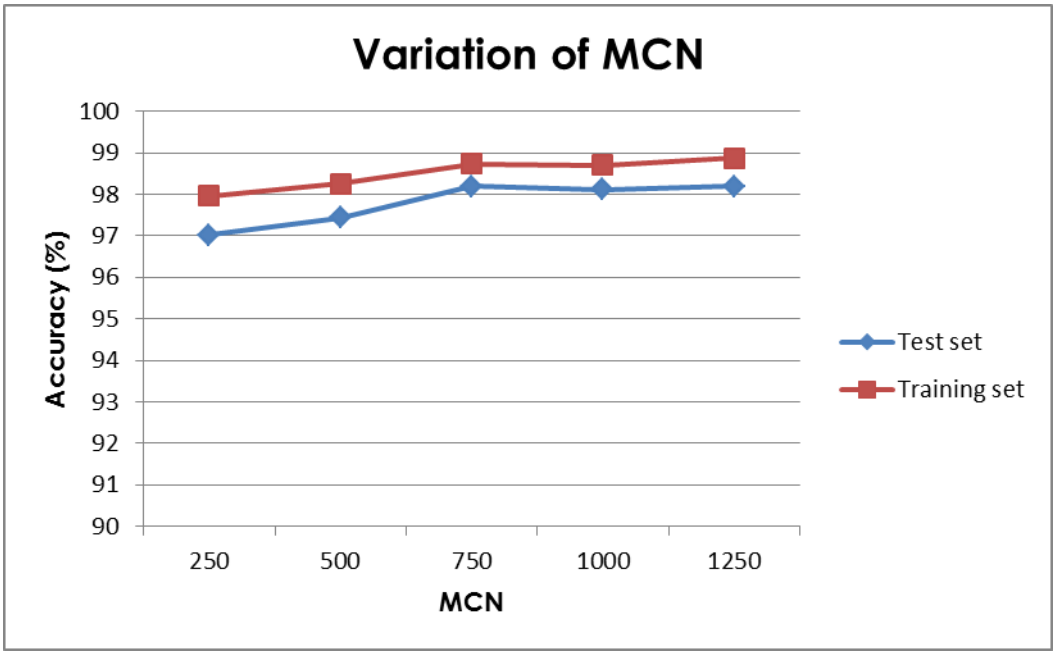


Figure 6.6: Mean accuracy when varying the number of cycles

Aside from varying the ABC parameters, we also conducted an experiment varying the number of hidden nodes of the neural networks. Shown in Table 6.17 are the time results of the five setups. We can say that the training time is directly proportional to the number of hidden nodes.

Since the hidden layer is located in between the input layer and output layer, there is a significant change in the total number of interconnections in the network if the number of hidden nodes is changed. These interconnections are the ones being optimized in the neural networks which require information to flow among nodes through their

interconnections. In other words, each connection represents a single parameter in the solution. If there are more hidden nodes, then the dimension of the solutions gets larger. So there will be more parameters that need to be optimized resulting to more processing time.

Table 6.17: Training time results (in seconds) when varying the number of hidden nodes

	A	B	C	D	E
Mean	106.42	122.30	153.46	217.57	275.27
Median	106.63	121.42	151.77	214.81	274.76
Maximum	114.36	135.90	173.32	244.57	286.21
Minimum	98.91	117.58	146.10	209.88	268.87
SD	3.533241	4.036018	5.873779	7.804307	3.675536
Variance	12.48379	16.28944	34.50128	60.90721	13.50956

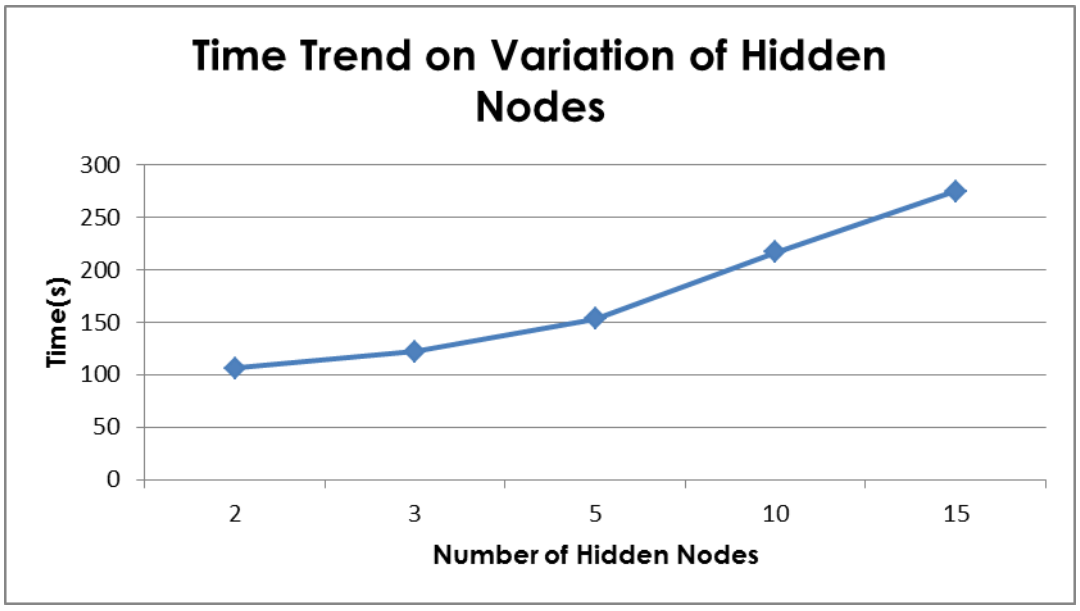


Figure 6.7: Average training time on varying the number of hidden nodes

Experiment results show that having two (2) and three (3) hidden nodes are not enough for the ABC-NN classifier to perform well both in the test set and training set. Large values for the standard deviation and variance can be observed for these two

setups. The other setups which have at least as many nodes as the input layer and output layer got good results.

Hidden nodes give the neural network the ability to generalize in complex problems. But very few numbers of hidden nodes will have difficulty generalizing such problems. This is true in this research as the success rate is low when using only 2 or 3 hidden nodes. Increasing the number of hidden nodes provides more flexibility to the neural network allowing it to perform a bit better.

Table 6.18: Test set results when varying the number of hidden nodes

	A	B	C	D	E
Mean	83.80 %	93.15 %	98.19 %	98.39 %	98.61 %
Median	83.89 %	95.00 %	98.34 %	98.89 %	98.33 %
Maximum	97.22 %	98.33 %	100.0 %	100.0 %	100.0 %
Minimum	65.00 %	77.22 %	96.11 %	96.11%	96.11 %
SD	8.530968	4.857118	1.041224	0.915198	0.896398
Variance	72.77742	23.59160	1.084147	0.837587	0.803531

Table 6.19: Training set results when varying the number of hidden nodes

	A	B	C	D	E
Mean	84.52 %	94.09 %	98.73 %	99.37 %	99.35 %
Median	83.81 %	95.12 %	98.81 %	99.29 %	99.29 %
Maximum	97.38 %	98.57 %	99.52 %	99.52 %	99.52 %
Minimum	65.71 %	79.76 %	96.90 %	99.05 %	99.05 %
SD	7.913413	4.540021	0.620035	0.169324	0.164641
Variance	62.62211	20.61180	0.384443	0.028670	0.027106

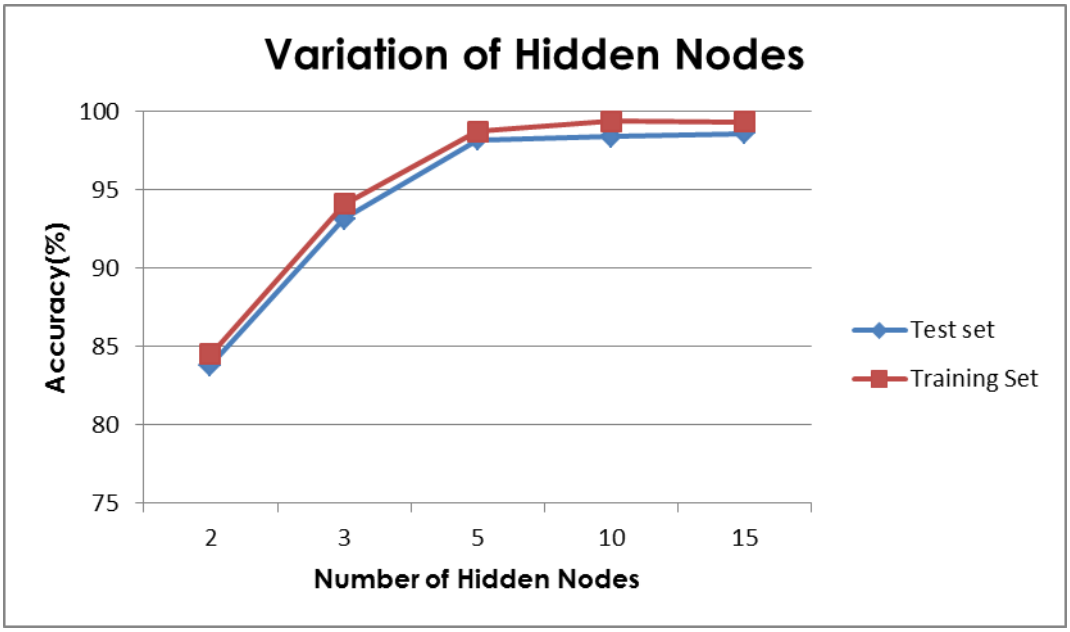


Figure 6.8: Mean accuracy when varying the number of hidden nodes

For the last experimentation, we varied the input color features to see its effect in tomato classification. Table 6.20 shows the experimental setups with different color features. The first three setups are based on the previous researches while the fourth setup includes the proposed color features used in this paper.

Table 6.20: Varying the input color features

Set	Color Feature/s
A	**a***
B	**R-G, Hue**
C	**Red, Green, Blue**
D	**Red, Green, Red-Green, Hue, a***

Since each color feature corresponds to an input node, the number of input nodes and the dimension of the neural networks varies in each setup. Similar to the previous experiment, increasing the number of input nodes increases the dimension of the solutions, thus increases the training time. Table 6.21 and Figure 6.9 show the trend in training time.

Table 6.21: Training time results (in seconds) when varying the input color features

	A	B	C	D
Mean	133.04	139.69	144.14	153.46
Median	132.34	139.97	142.46	151.77
Maximum	145.44	144.00	156.30	173.32
Minimum	128.31	134.69	137.94	146.10
SD	3.844742	2.507059	5.709861	5.873779
Variance	14.78204	6.285346	32.60251	34.50128

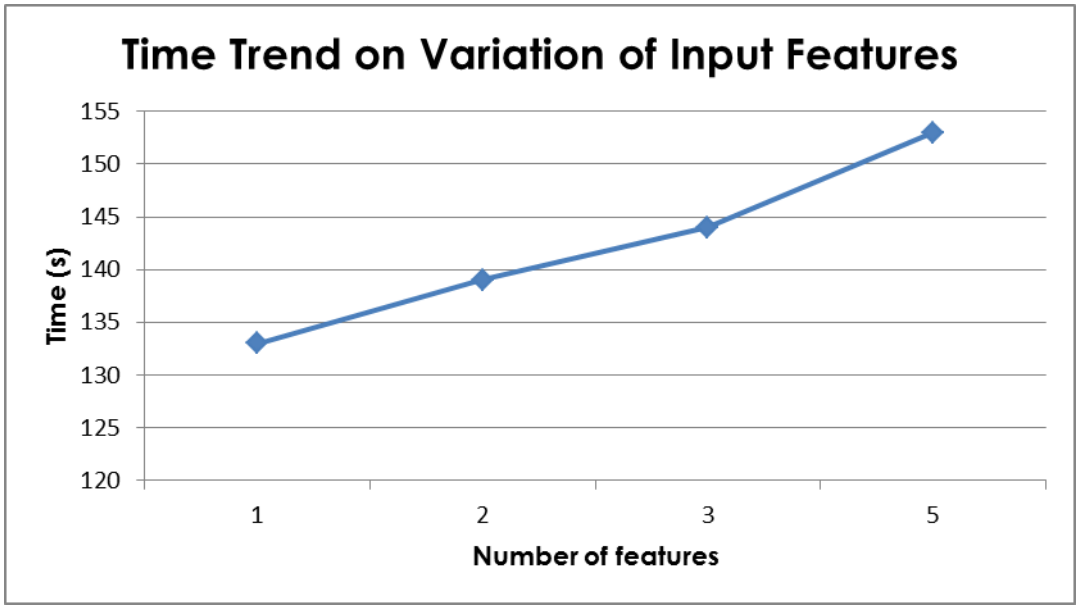


Figure 6.9: Average training time on varying the number of input features

Table 6.22 and Table 6.23 show that set A and set C, which used a single color model, produced the lowest results. Also, increasing the number of input features does not necessarily increase the accuracy as set B (2 color features) got better results than set C (3 color features). These observations are true in both the test set and training set.

Although a* alone performed well in the previous researches, it is not the case in this research with a large dataset. Because of the large dataset, it is very difficult to model tomatoes with six maturity stages using a single color feature. There is a high chance that

a* values overlap especially in the middle stages causing the neural network to misclassify some data.

Red, Green, and Blue elements performed better than a* alone because of the fact that three features are better than one. But this is not the case when we compare set B and C. According to [16], RGB color model is affected by illumination. With our large dataset, it is very difficult to have uniform illumination for all tomato images. In effect, RGB color values fluctuate among similar maturity stages. Unlike a single RGB element, R-G color feature appears to be stable across different stages. R-G combined with Hue color element produced better results than R, G, and B elements. With these observations, no wonder why combining five color elements (R, G, R-G, Hue, and a*) got the best results with very high accuracy rate and very small values for the standard deviation and variance.

Table 6.22: Test set results when varying the input color features

	A	B	C	D
Mean	83.20 %	95.39 %	92.93 %	98.19 %
Median	82.22 %	95.56 %	93.33 %	98.34 %
Maximum	90.55 %	97.22 %	97.78 %	100.0 %
Minimum	82.22 %	92.22 %	77.22 %	96.11 %
SD	2.291467	1.430599	3.917338	1.041224
Variance	5.250821	2.046615	15.34554	1.084147

Table 6.23: Training set results when varying the input color features

	A	B	C	D
Mean	83.24 %	96.33 %	94.25 %	98.73 %
Median	82.38 %	96.67 %	94.88 %	98.81 %
Maximum	89.76 %	98.10 %	98.57 %	99.52 %
Minimum	81.90 %	91.90 %	80.24 %	96.90 %
SD	2.139189	1.408729	3.742082	0.620035
Variance	4.576132	1.984518	14.00318	0.384443

Chapter 7

Conclusion and Future Works

In this paper, an Artificial Bee Colony algorithm trained Artificial Neural Network classifier was implemented to solve the tomato maturity classification problem. Experimental results show that the classifier performed efficiently both in the test set and training set. To further verify its efficiency an experiment was conducted that varies the test set and training set in each run. This experiment also produced satisfactory results.

Variation of the ABC and NN parameter values is one of this research's objectives. It can be easily observed that the training time is directly proportional to the parameter values. In terms of accuracy, there was an obvious drop in the success rate when the number of the employed bees and hidden nodes are too small. Since the size of the solution space depends on the number of employed bees, it is important to keep this value to a certain minimum. On the other hand, the number of hidden nodes cannot be too small since it is a core component in a multi-layer perceptron. Varying the number of onlooker bees and MCN did not show significant trends.

Moreover, there is a significant effect in the success rate when the input color features are varied. A single color feature as input to the neural network does not guarantee a good result because of the closeness of the values among different maturity stages. But increasing the number of input features does not necessarily improve the success rate such as when using the RGB color model alone. Instead, choosing the right color features as inputs the neural network should be taken into consideration. In this paper, the combination of five color features (R, G, R-G, Hue, and a*) from three color models (RGB, HSI, CIE L*a*b*) has successfully classified tomato based on the USDA tomato color chart.

The experimental results of this study are very satisfactory and we can conclude that the ABC-NN classifier is an efficient tool to automate tomato maturity classification.

Future works could involve an online tomato maturity grading system that uses some hardware tools that this research could not avail. Also, the optimization algorithm

can be improved or replaced by other relatively new algorithms such as Cuckoo Search and Firefly Algorithm. Furthermore, the same approach can be applied to determine the level of maturity of other fruits such as apples and mangoes.

References

[1] Alimi, A. M., Dhari, H.: Designing Beta Basis Function Neural Network for Optimization Using Artificial Bee Colony (ABC), IEEE World Congress on Computational Intelligence, (2012)

[2] An Introduction to Image Processing. From, http://bit.ly/Ud92qI (Accessed: July, 2012)

[3] Artificial Bee Colony Algorithm. From, http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm (Accessed: July, 2012)

[4] Baek, I-S., Cho, B-K., Kim, Y-S.: Development of a Compact Quality Sorting Machine for Cherry Tomatoes Based on Real-Time Color Image Processing, CIGR-Ageng, (2012)

[5] Baik, S. W., Kim, W., Lee, H-K., Yoo, S. J.: Neural Network Based Adult Image Classification, ICANN 2005, LNCS 3696, (2005). 481 – 486

[6] Binary Image Processing. From http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/medium/binary/index.html (Accessed: October 2012)

[7] Chen, M.: Analysis and Design of the Multi-Layer Perceptron using Polynomial Basis Functions (Unpublished) (1991)

[8] Choosing a Color Space. From http://www.oppodigital.com/blu-ray-benchmark/blu-ray-benchmark-Choosing_a_Color_Space.asp (Accessed: July 2012)

[9] Chung, Y. K., Shukran, M. A. M., Wahid, N., Yeh, W-C., Zaidi, A. M. A.: Artificial Bee Colony based Data Mining Algorithms for Classification Tasks, Canadian Center of Science and Education, Vol. 5, No. 4, (2011)

[10] CIELAB. From http://dba.med.sc.edu/price/irf/Adobe_tg/models/cielab.html (Accessed: July 2012)

[11] Color Space. From http://www.colorbasics.com/ColorSpace/ (Accessed: July 2012)

[12] Connsynn, C., Suryanti, A., Syahrir, W. M.: Color Grading in Tomato Maturity Estimator using Image Processing Technique, IEEE International Conference on Computer Science and Information Technology, (2009)

[13] Fundamentals of Image Processing. From, http://bit.ly/XjB1YP (Accessed July 2012

[14] Fundamentals of Neural Networks. From http://www.myreaders.info/08_Neural_Networks.pdf (Accessed: August 2012)

[15] Gejima, Y., Ishino, F., Nagata, M., Tallada, J.: Estimation of Tomato Ripening Stages Using Three Color Models, Bulletin of the Faculty of Agriculture, Miyazaki University, vol. 50(1·2), (2004) 65-71

[16] Gejima, Y., Nagata, M., Zhang, H.: Judgment on Level of Maturity for Tomato Quality Using L*a*b* Color Image Processing, IEEE/ASME International Conference on Advanced Intelligent Machatronics, (2003)

[17] Ghazil, R., Nawi, N. M., Shah, H.: Using Artificial Bee Colony Algorithm for MLP Training on Earthquake Time Series Data Prediction (Unpublished) (2011)

[18] Grayscale. From http://whatis.techtarget.com/definition/grayscale (Accessed: August 2012)

[19] Han, X., Mao, H., Wang, X., Yin, J.: Vision-based judgment of tomato maturity under growth conditions, African Journal of Biotechnology, vol. 10(18), (2011) 3616-3623

[20] How to Grow Tomatoes. From http://www.gardenersnet.com/vegetable/tomato.htm (Accessed: July 2012)

[21] HSI Color Space – Color Space Conversion. From http://www.scribd.com/doc/16617255/HSI-Color-Space (Accessed: July 2012)

[22] Hussain, A., Saad, H.: Classification for the Ripeness of Papayas Using Artificial Neural Network (ANN) and Threshold Rule, 4th Student Conference on Research and Development (SCOReD 2006), (2006)

[23] Identification of trichomes, loci and chemical compounds derived from Solanum habrochaites accession LA1777 that are associated with resistance to the sweet potato whitefly, Bemisia tabaci in tomato, S. lycopersicum. From http://tgc.ifas.ufl.edu/Presentations/7%20Mohamed%20Breeding-some%20changes.pdf (Accessed: July 2012)

[24] Image Processing IDL. From http://bit.ly/ZopyJE (Accessed: September 2012)

[25] Intelligent systems and Neural Networks. From http://www.learnartificialneuralnetworks.com/ (Accessed: August 2012)

[26] Introduction to Image Processing. From http://www.engineersgarage.com/articles/image-processing-tutorial-applications (Accessed: August 2012)

[27] Kamatis. From http://www.stuartxchange.org/Kamatis.html (Accessed: September 2012)

[28] Karaboga, D.: An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, (2005)

[29] Karaboga, D., Ozturk, C.: Fuzzy clustering with artificial bee colony algorithm, Scientific Research and Essays, vol. 5(14), (2010) 1899-1902

[30] Krishnan, S., Kumbhar, P. Y.: Use Of Artificial Bee Colony (ABC) Algorithm in Artificial Neural Network Synthesis, IJAEST, vol. 11, issue 1, (2011) 162 – 171

[31] May, Z., Amaran, M. H.: Automated Oil Palm Fruit Grading System using Artificial Intelligence, International Journal of Video & Image Processing and Network Security IJVIPNS-IJENS, vol. 11, no. 3, (2011)

[32] McCarthy, M. J., Zhang, L.: Measurement and evaluation of tomato maturity using magnetic resonance imaging. Postharvest Biology and Technology 67, (2011) 37-43

[33] MLP. From http://www.dtreg.com/mlfn.htm (Accessed: September 2012)

[34] Neural Networks. From http://library.thinkquest.org/C007395/tqweb/history.html (Accessed: August 2012)

[35] Otsu Thresholding. From http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html#explained (Accessed: October 2012)

[36] RGB Channels - Mike's Sketchpad. From http://www.sketchpad.net/channels1.htm (Accessed: July 2012)

[37] Seven Grayscale Conversion Algorithms - Tanner Helland. From http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/ (Accessed: August 2012)

[38] The RGB (CMY) Color Model. From http://dba.med.sc.edu/price/irf/Adobe_tg/models/rgbcmy.html (Accessed: July 2012)

[39] Tomatoes. From http://www.whfoods.com/genpage.php?tname=foodspice&dbid=44 (Accessed: July 2012)

[40] University of California Agriculture and Natural Sciences. From, http://ucanr.edu/repository/view.cfm?article=83755 (Accessed: July 2012)

[41] Wang, S., Wu, L., Zhang, Y.: Magnetic Resonance Brain Image Classification by an Improved Artificial Bee Colony Algorithm, Progress in Electromagnetics Research, vol. 116, (2011) 65-79

APPENDIX

USER'S MANUAL

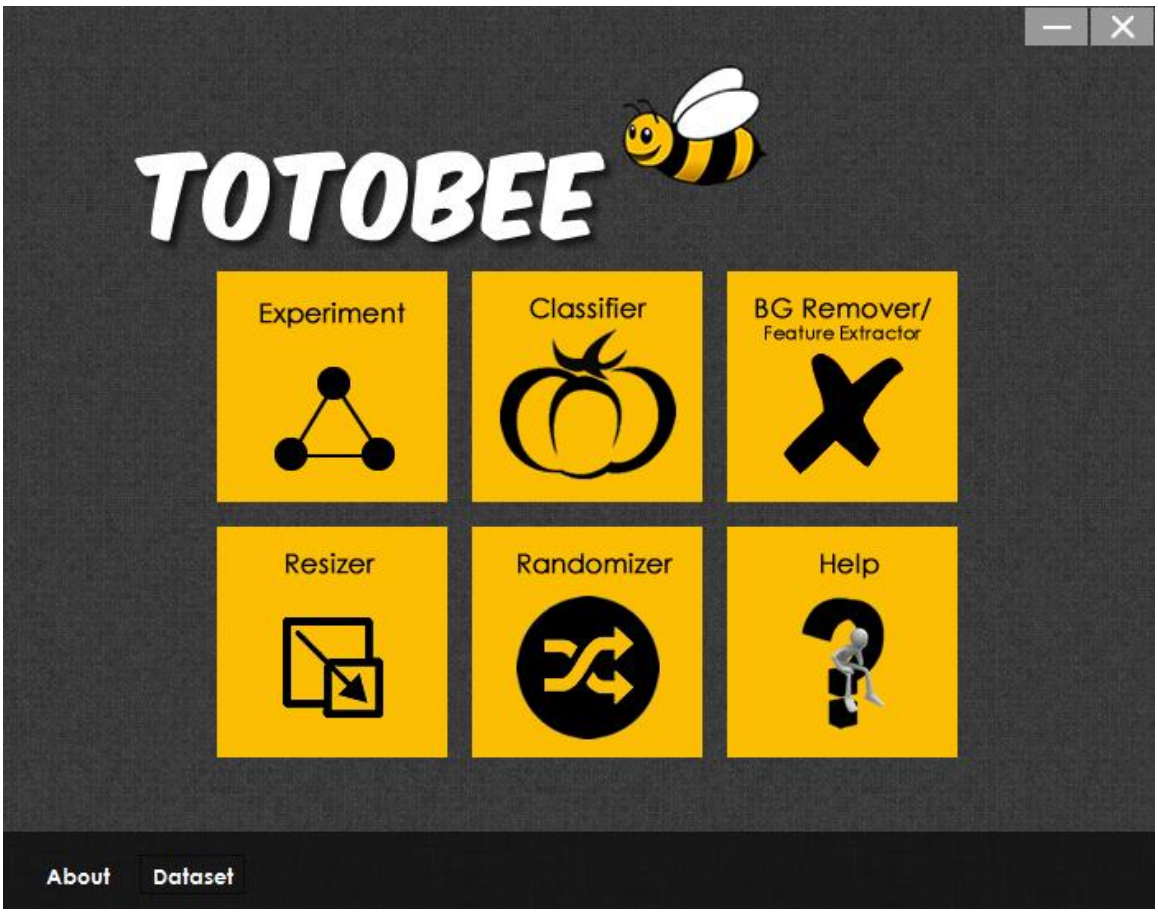
Splash Screen

TotoBee's User Interface

The program's user interface includes six modules:

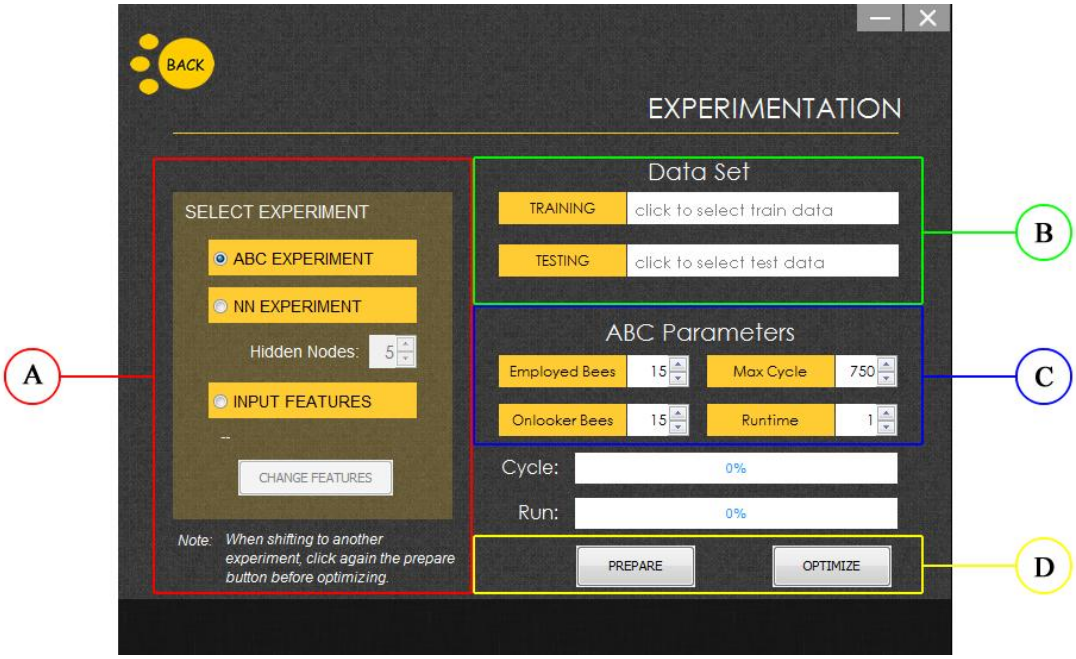
- Experiment
- Classifier
- Background Remover / Feature Extractor
- Resizer
- Randomizer
- Help

Aside from the six main buttons, there are also two buttons at the bottom of the interface - the **About** and **Dataset** buttons.



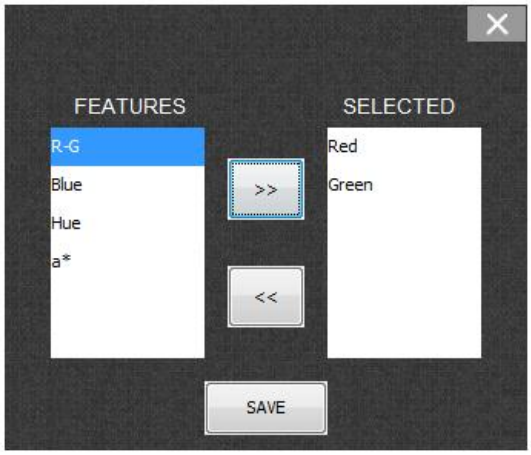
Experiment Module

The Experiment Module is subdivided into four panels shown below:



A. Experiment Selection

- Allows the user to select type of the experiment. The default experiment is the ABC Experiment wherein the user can alter the values of the ABC parameters in the ABC Parameters panel.
- If the user selects the Input Features Experiment, he can choose the features he wants to use by clicking the **CHANGE FEATURES** button then it will show a dialog like the one below:



B. Data Set

- The dataset consists of the training set and test set. The training set is used to train the Artificial Neural Networks while the test set is used to test the effectiveness the optimal ANN. Training set and test set are files with a .data file extension.

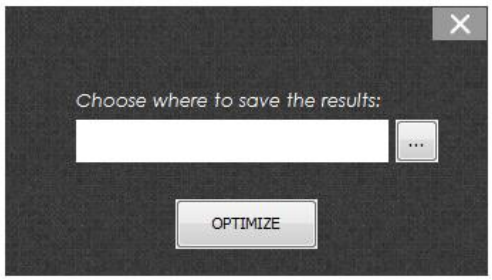
C. ABC Parameters

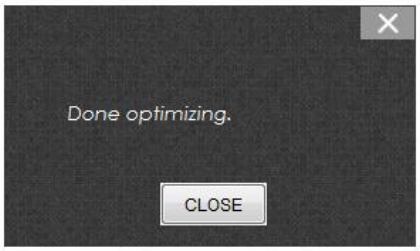
- Allows the user to change the parameter values of the ABC algorithm to be used in training the ANN.

D. Controls

- **PREPARE** Button
  - Used to load the training data as well the test data which are images of tomatoes. After loading the data, the images will undergo image preprocessing the feature extraction. A dialog will show once preparing is done.

- **OPTIMIZE** Button
  - Clicking the optimize button will show a dialog asking where to save the results of experimentation. If the user wishes to cancel the process, he can just close the dialog by clicking the **X** button. A dialog will show when the optimization ends.



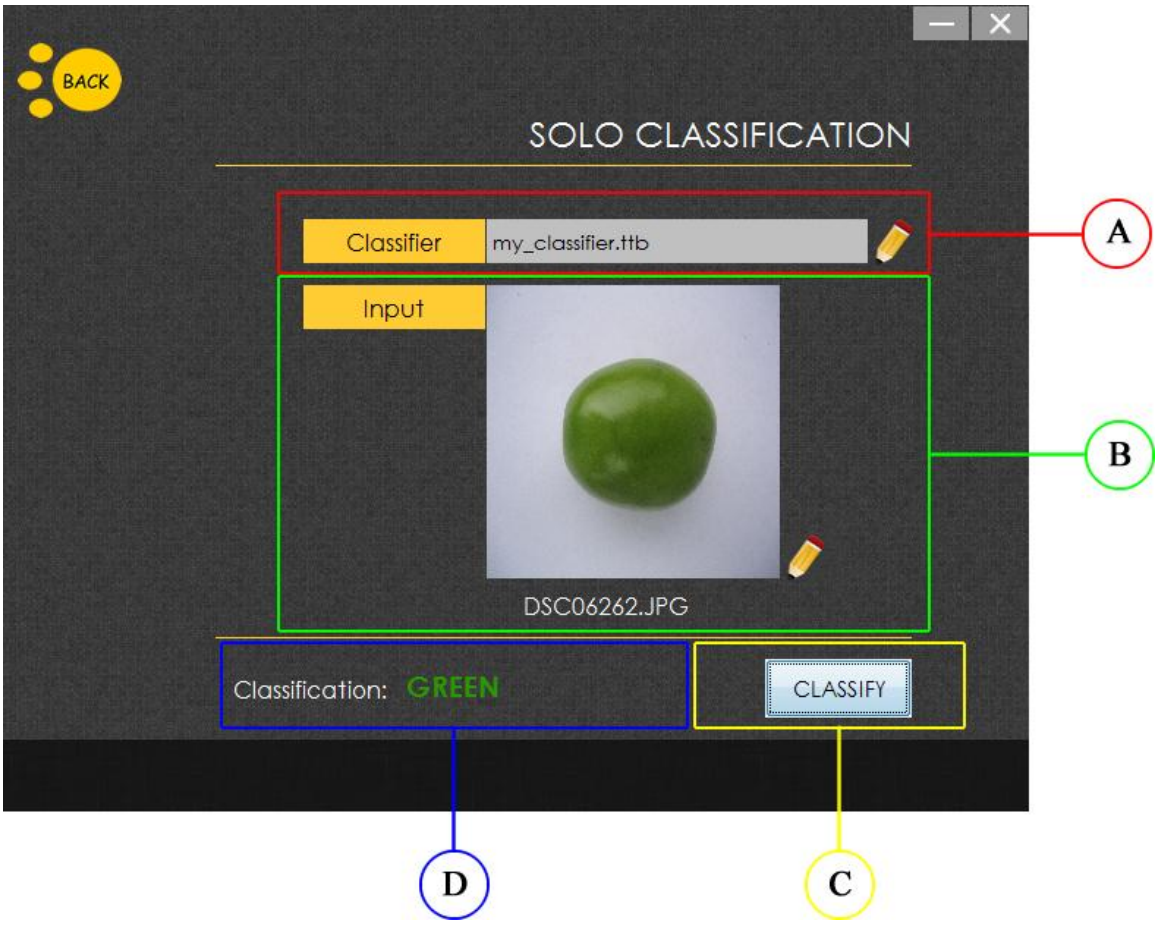


Classifier Module

The classification can be by batch or solo. A dialog will show letting the user select which type of classification he wants.

- **Solo Sub-Module**

  Allows the user to classify a single tomato image. The Solo sub-module is subdivided into four panels as shown below.



A. Classifier

- The user can select which classifier he wants to use. Classifiers are files with a .ttb file extension. They contain solutions generated from training the ANN in the Experiment Module.

B. Input Image

- Allows the user to select an input image to be classified.

C. Classify Button

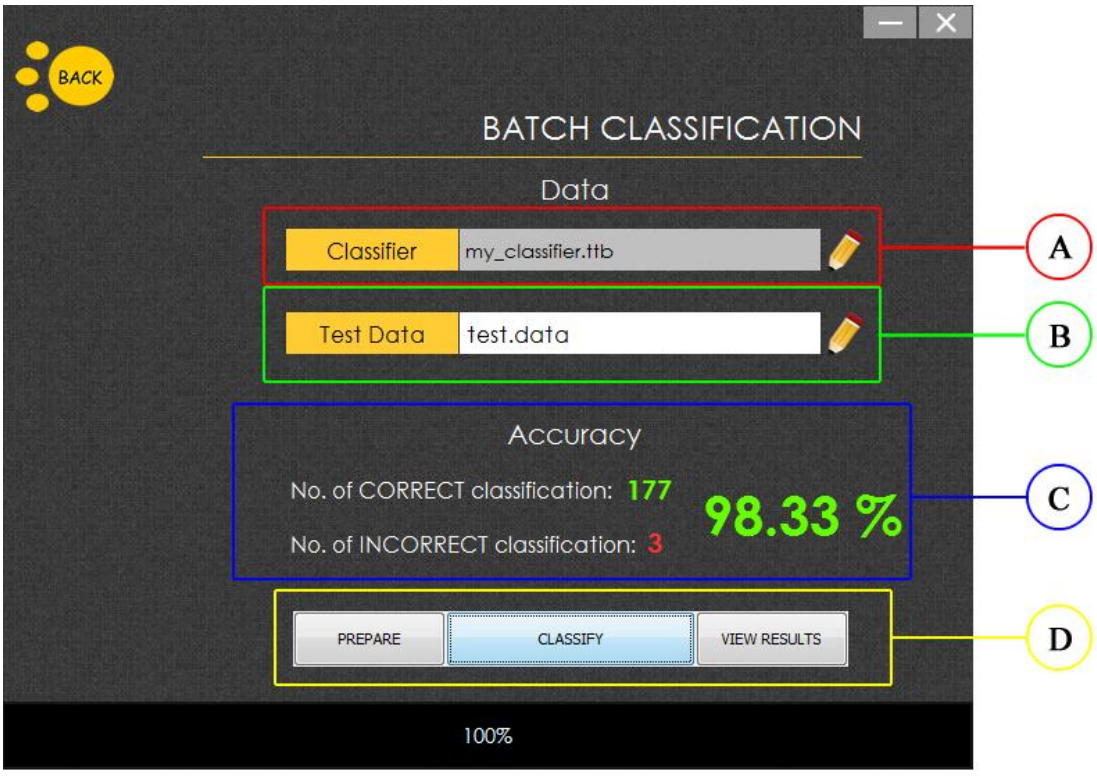
- Clicking this button will start the classification.

D. Classification Result

- Displays the maturity stage of the input image.

- **Batch Sub-Module**

Allows the user to classify multiple tomato images. The figure below shows the look of this sub-module.



A. Classifier

- This is the same with the one in the previous sub-module.

B. Input Data

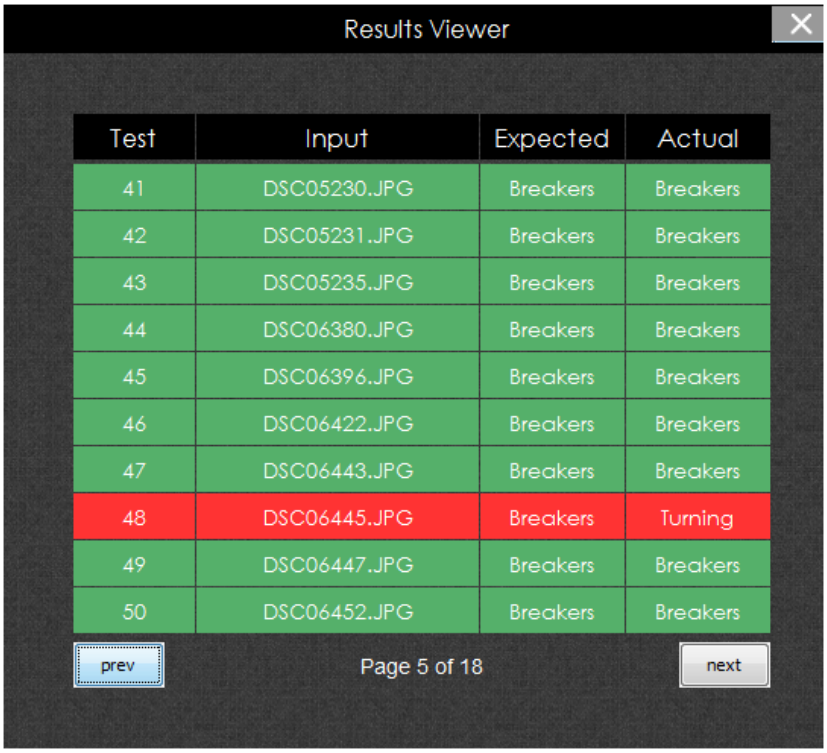
- Allows the user to select the data to be classified. These are files with a .data file extension. This can be the same with the .data used in the Experiment module.

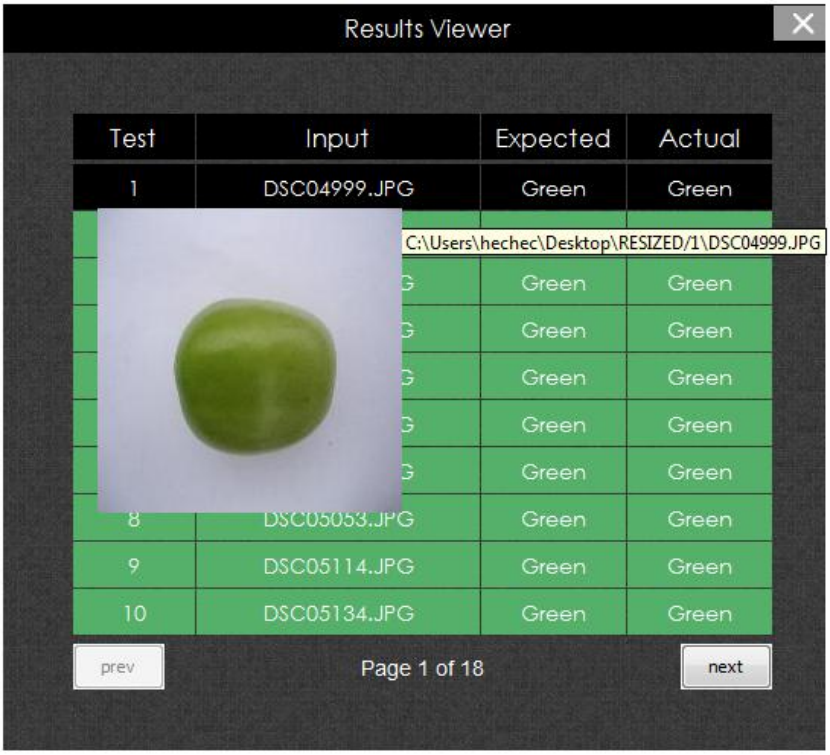
C. Results Information

- Displays the results of the classification. This includes the number of correctly classified, incorrectly classified tomatoes, and the classification's accuracy.

D. Controls

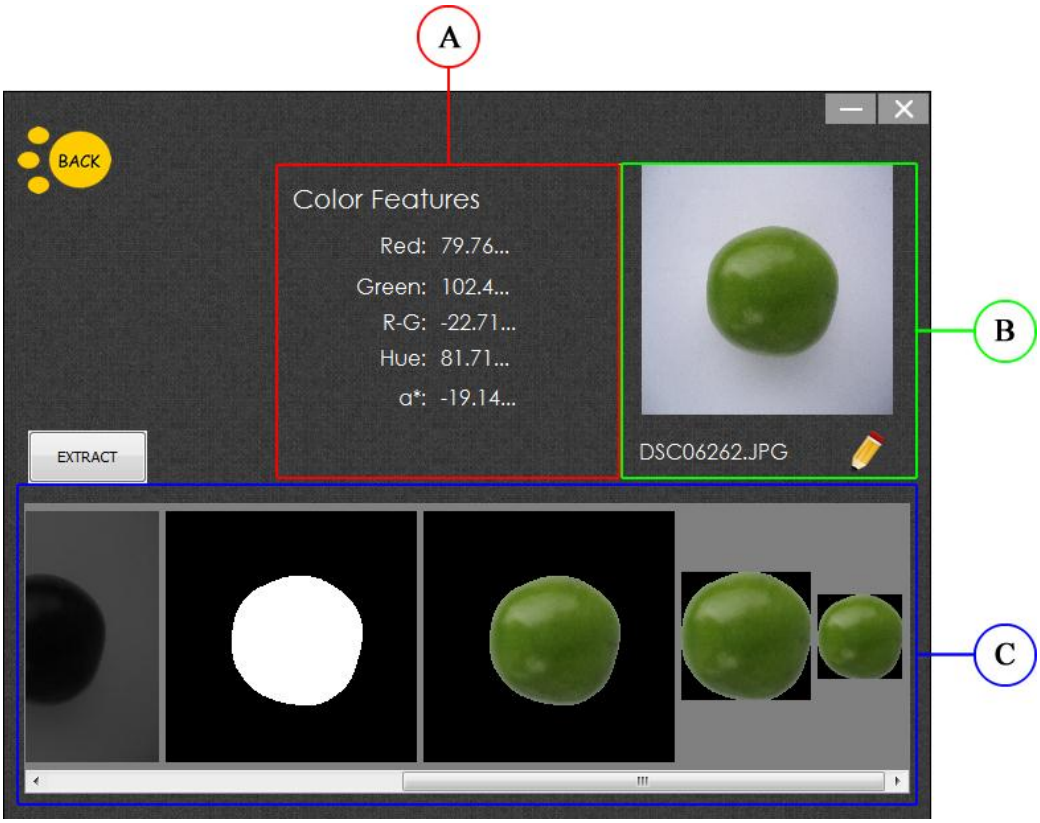
- **PREPARE** Button
  - This is the same with the **PREPARE** button in the Experiment module.
- **CLASSIFY** Button
  - Clicking this button will start the classification.
- **VIEW RESULTS** Button
  - Clicking this button will show a dialog box containing the detailed results of the classification.
  - Hovering a specific row on the results viewer will show the image of the tomato and the absolute path of the image.


Results Viewer
Test
Input
Expected
Actual
41 DSC05230.JPG Breakers Breakers
42 DSC05231.JPG Breakers Breakers
43 DSC05235.JPG Breakers Breakers
44 DSC06380.JPG Breakers Breakers
45 DSC06396.JPG Breakers Breakers
46 DSC06422.JPG Breakers Breakers
47 DSC06443.JPG Breakers Breakers
48 DSC06445.JPG Breakers Turning
49 DSC06447.JPG Breakers Breakers
50 DSC06452.JPG Breakers Breakers
prev
Page 5 of 18
next


Results Viewer
Test
Input
Expected
Actual
1 DSC04999.JPG Green Green
C:\Users\hechec\Desktop\RESIZED/1\DSC04999.JPG
Green Green
Green Green
Green Green
Green Green
Green Green
8 DSC05053.JPG Green Green
9 DSC05114.JPG Green Green
10 DSC05134.JPG Green Green
prev
Page 1 of 18
next

Background Remover / Feature Extractor Module

This module allows the user to visualize the different image processing steps to remove the image background, and the actual color features extracted from the input image. The figure below shows three panels from this module.



A. Color Features

- Shows the color features extracted from the input image.

B. Input

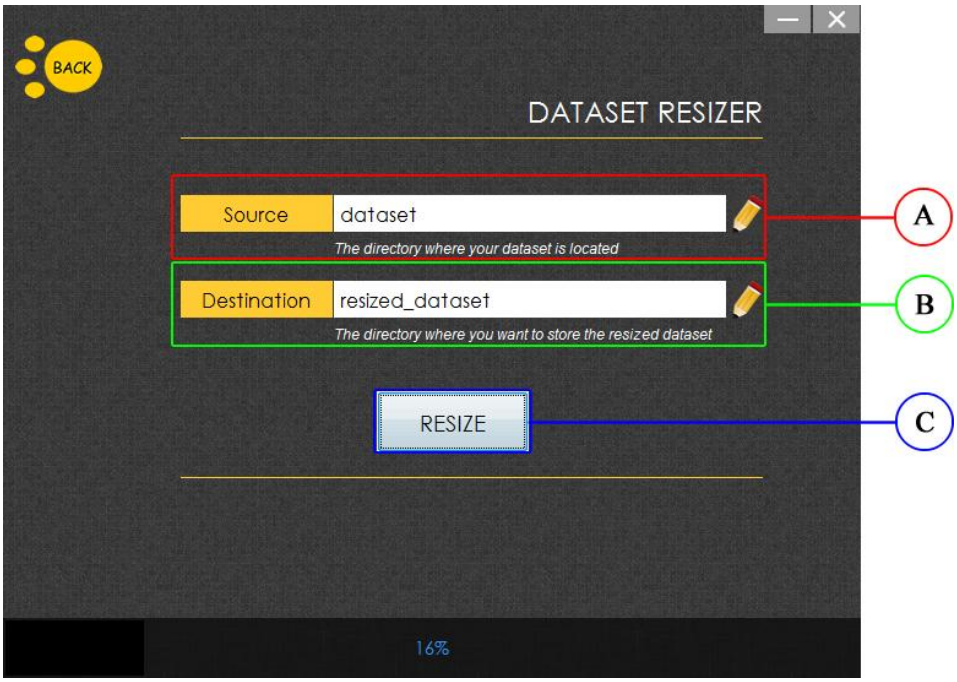
- Allows the user to select an image for preprocessing the feature extraction.

C. Image Processing

- Shows a series of image processing steps needed before extracting the color features

Resizer Module

This module allows the user to resize the dataset to 200 x 200 pixels since the original size of the images are very large. Given a folder/directory, this module will resize all images inside the subfolders.



A. Source

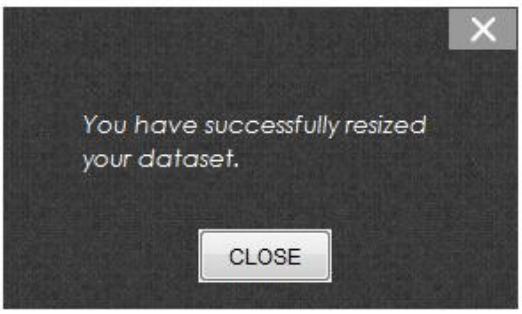
- Allows the user to select the location of the dataset to be resized

B. Destination

- Allows the user to select the location of the resized dataset

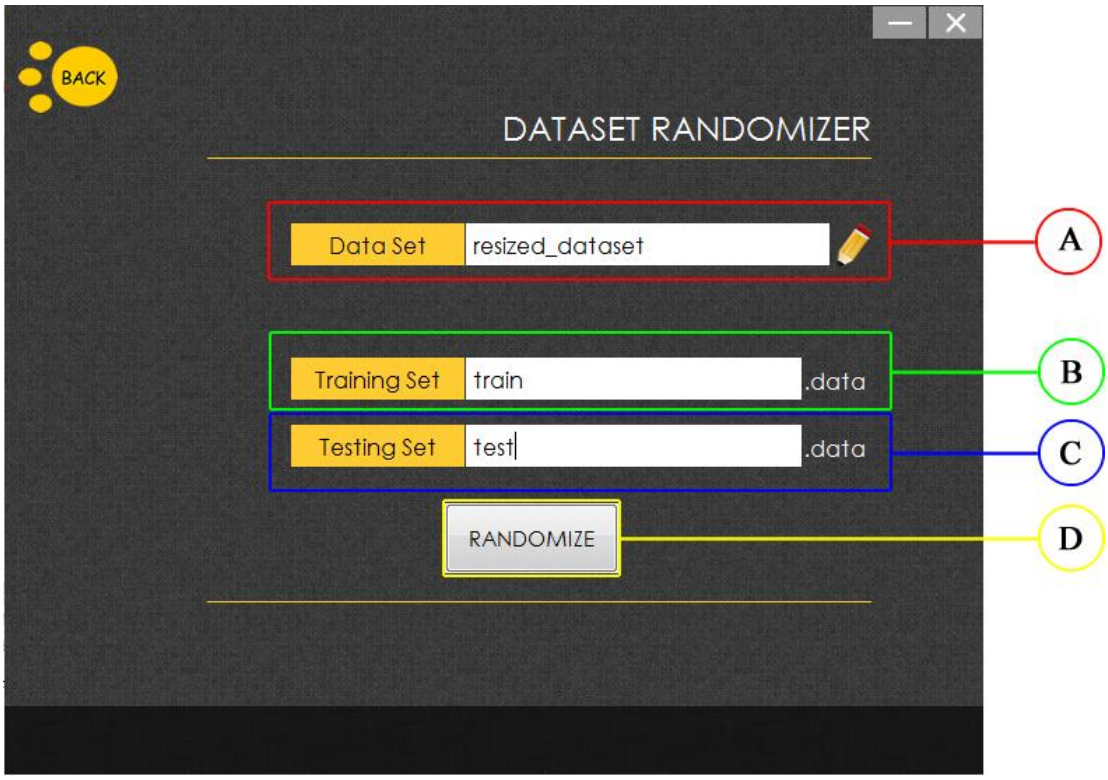
C. Resize Button

- Clicking the **RESIZE** button start the resizing. A dialog will show if the resizing is done.



Randomizer Module

This module allows the user to randomize the dataset (training set and test set). The dataset will be divided seventy percent (70%) to the training set and thirty percent (30%) to the test set. Both data sets will be saved as .data.



A. Dataset

- Allows the user to browse the location of the dataset.

B. Training Set

- Allows the user to give a filename for the training set

C. Test Set

- Allows the user to give a filename for the test set

D. Randomize Button

- Clicking the **RANDOMIZE** button will start randomizing the dataset
- A dialog will show asking where to save the randomized training set and test set
- Also, a dialog will show when the program finished randomizing.

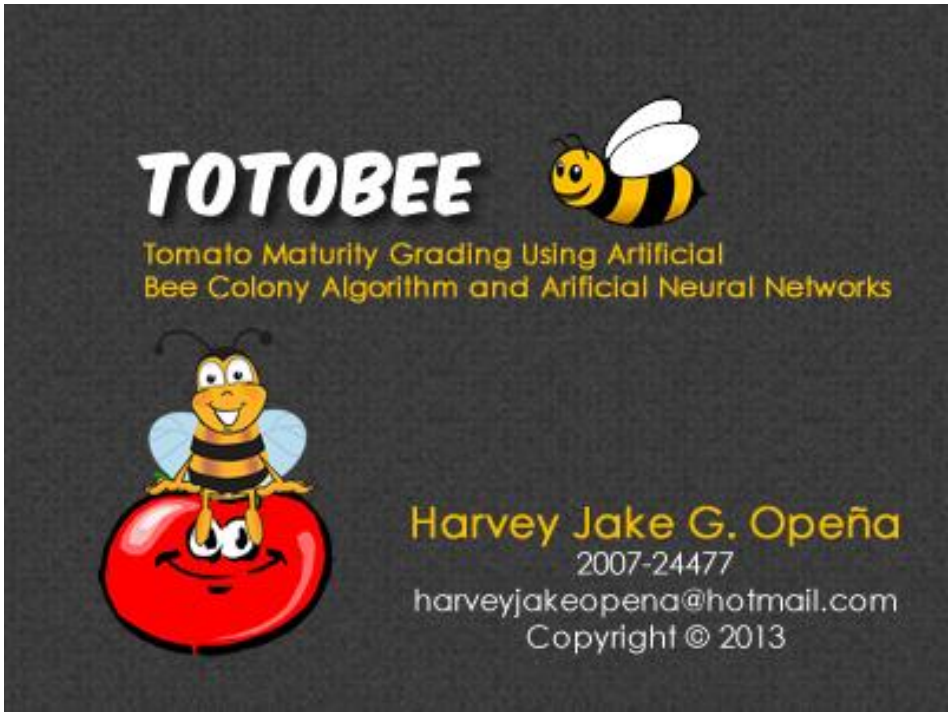
Help Module

Provides a User's Manual on how to use the TotoBee application.

Dataset

This allows the user to locate the dataset to be used since there could be many datasets located anywhere in the file system. The selected dataset directory should contain the files listed in the .data files to be used in the experiments and classification.

About


TOTOBEE
Tomato Maturity Grading Using Artificial
Bee Colony Algorithm and Arificial Neural Networks
Harvey Jake G. Opeña
2007-24477
harveyjakeopena@hotmail.com
Copyright © 2013