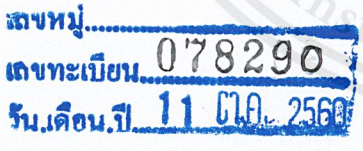
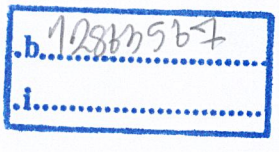
Plant Nutrient Deficiency Classification System using Convolutional Neural Networks

Pavit Noinongyao
Chaiwat Wattanapaiboonsuk
Puriwat Khantiviriya





Bachelor of Engineering in Software Engineering,
International College,
King Mongkut's Institute of Technology Ladkrabang,
Academic Year 2016
KMITL-2017-IC-B-003-004

Thesis - Academic Year 2016

Bachelor of Engineering in Software Engineering

International College

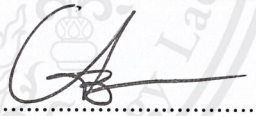
King Mongkut's Institute of Technology Ladkrabang

Title: Plant Nutrient Deficiency Classification System

Authors:

1. Pavit Noinongyao Student ID: 56090015
2. Chaiwat Wattanapaiboonsuk Student ID: 56090038
3. Puriwat Khantiviriya Student ID: 56090044

Approved for Submission



.....

(Dr. Ukrit Watchareeruetai)

Advisor

Date 7/6/2017

Abstract

Pavit Noinongyao 56090015
Chaiwat Wattanapaiboonsuk 56090038
Puriwat Khantiviriya 56090044
Dr. Ukrit Watchareeruetai Advisor
Academic Year 2016

Keywords: convolutional neural network, image processing, image segmentation, black gram, leaf segmentation, plant nutrient deficiency detection, agriculture, active contour model, semi-automatic segmentation.

Nutrient deficiencies in plants cause a lot of problems. They lead to slow or even stuck in growth of plants and also the productivity. In agriculture, these problems are considered very severe. Therefore, an ability to robustly detect these deficiencies is extremely desirable.

Nutrient deficiencies also lead to unusual appearance on a plant's leaves and they can be observed visually. This thesis proposes a system for classifying deficiencies presented on a plant using its leaf image. The system comprises two main components which are mobile application and the server. The mobile application is used to acquire a leaf image and segment background region using active contour model and send the image to the server. The server receives and classifies the deficiency presented in the input image. To classify the input image, it is first divided into blocks of equal size without background region. Each block is fed into multiple convolutional neural networks (CNNs) implementing multilayer perceptron-based one-vs-all strategy to produce block-level labels. The portions of each label are used to feed to another multilayer perceptron to determine the final nutrient deficiency of the whole leaf image. The result is then send back to the mobile application

Experimental results show that on average, the system outperform human ability in recognizing plant nutrient deficiencies on leaf images.

Acknowledgments

During the development of the proposed system, Plant nutrient deficiency classification system, We had done a lot of works and researches. Many problems had arisen during this time. Without considerable supports given to us, the project will never be done. With these, we would like to dedicate this part of the thesis to express our deepest appreciation to all of them.

First of all, we would like acknowledge our project advisor Dr. Ukrit Watchareeruetai, for strengthening our specialization, and providing us a place, equipment and technical support in learning the subject. Without his sincere devotion, we would not be able to reach our goal in completing this project.

We are also grateful to, Asst. Prof. Dr. Chaiwat Nuthong, Dr. Visit Hirankitti, Dr. Isara Avantavrasilp, for criticizing and suggesting a way to improve our project and presentation. Ultimately, we would to express our thankful onto our families, seniors in computer vision lab and other colleagues for every kindness of cooperation, suggestion along with keeping us motivated and giving us strength to finally complete our project. Any errors that might arise in this report, it is due to the authors' mistakes.

Table of Contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation and problem description

Plants are growing up everywhere around the world and human uses them as food, seasonings, and medicines. In the history of mankind, we normally do agriculture to grow and harvest plants. They are very important to human life, especially in Thailand where its economy relies heavily on agriculture.

Diseases in plants are one of the major problems which lead to losses in agricultural economy, and by being able to identify the symptom of infected plants beforehand, we can significantly reduce the risk of economic loses.

Diseases in plants might be caused by pests, bacteria, virus, or nutrient deficiencies. Those are external factors but nutrient deficiencies are internal, which are harder to identify than the external one. This abnormality leads to much unusual symptom in plants.

Nutrient deficiencies in a plant can cause a plant to not bloom, fertilize or bear fruit properly. Fortunately, these deficiencies also lead to unusual appearance on the plant's leaves and can be detected visually and it will get more severe with time. However, it is quite difficult to identify them by naked eyes in the early stage and even after some period, it still requires an expert to identify the symptom. Different

deficiencies also cause different symptoms appeared on the leaves. Figure A.11 shows the an example of a healthy plant and Fig. A.12 shows an example of unusual appearance when a plant lacks some certain nutrients

Symptoms of deficiencies in each nutrient are summarized as follows:

- Lack of calcium (Ca): A leaf turns into yellow and white dots are presented in young leaves. The plant will grow slower than usual.
- Lack of iron (Fe): Young leaves turn into yellow and plant grow slower than normal because iron is very important in photosynthetic reactions.
- Lack of magnesium (Mg): Young and old leaves at the bottom of the plant contain yellow mottling apex brown, and brown mottling on the leaf margin.
- Lack of nitrogen (N): The plant has small pale green, pale yellow leaves. The symptom appears first in older leaves. Young leaves still have green color but the size is small.
- Lack of potassium (K): Yellowish-white mottling occur in the area close to the vein and the margin of a leaf. The leaf will turn into reddish-brown in the last stage. symptom occurs from the bottom to the top of the plant

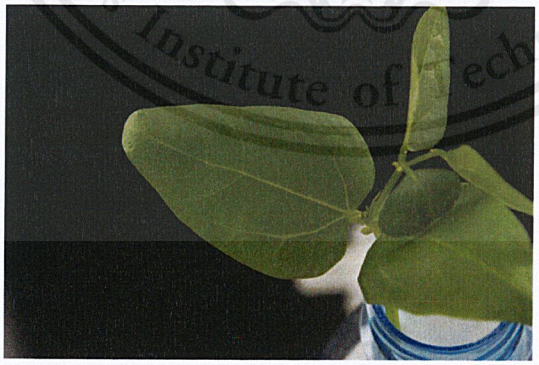
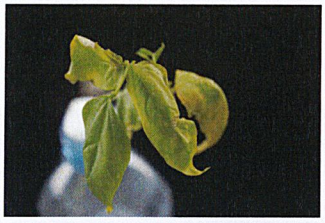


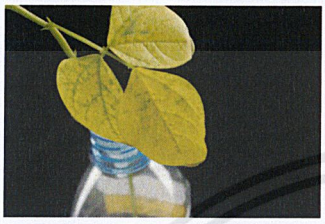
Figure 1.1: Examples of a healthy plant's leaf.



(a) Calcium deficiency

(b) Iron deficiency

(c) Magnesium deficiency



(d) Potassium deficiency

(e) Nitrogen deficiency

Figure 1.2: Examples of unusual appearance when a plant lacks some certain nutrients.

There are some methods used for nutrient deficiency identification and it can be categorized into two types which are destructive and non-destructive methods. Plant tissue analysis is one of the method used. It is a destructive method that requires a leaf sample from a plant in order to extract the nutrients by using a chemical process which takes long time. It also requires an expert to interpret the result. Furthermore, soil testing is another destructive method which uses a soil sample around the targeted plant instead of its leaf. The non-destructive method requires less physical contact and leaves no damage on a plant. For example, using plant nutrition analyzer is a non-destructive method. It is an instrument used to pinch on a leaf. The device then analyzes the deficiencies in the plant. The disadvantage of this method is that it can only detect the deficiencies in nitrogen and chlorophyll [1].

Another non-destructive method is to use computer vision. In computer vision, only images of a plant's leaf are required. In other words, computer vision approach visually identifies the deficiencies. Visual features of a leaf are used to classify it to be in each class of deficiencies. The features can be manually designed or

automatically built from training. The former requires a domain expert to select the features to be used for the classification to be effective. The latter requires no expertise but needs a lot of training example to train the machine. To classify the plants, various machine learning algorithms are used. However, it is still in the early stage of the research, thus, there are some proposed methods related to this, but the achieved accuracy is still quite far from practical. There are several related works concerning computer vision approach to plant disease identification which will be discussed next.

In 2015, Mokhtar et al. [14] developed plant disease detection in tomatoes using Gabor wavelet transform to extract relevant features in a tomato's leaf image. Then, they use support vector machine (SVM) with different kernel functions to detect and identify diseases on the plant. Kernel functions used in SVM are Cauchy function, Invmult function, and Laplacian function. In 2016, Rangel et al. [10] developed grapevine potassium deficiency diagnosis using image processing approach. The process is done by taking a leaf image and applying pre-processing to it. Then, pre-processed images are classified into three classes which belong to background, healthy tissues and reddish tissues. After that, they use information from healthy and reddish tissue classes to calculate Euclidean distance between points and class centroid and use the ratio between the total leaf area and the reddish tissue area for classification.

There are still many approaches that can be used to solve this kind of problem apart from the literature such as genetic programming and convolutional neural networks (CNNs) that might give better result. For example, CNNs which are proved to give an outstanding accuracy in other recognition tasks might also give good result in this field.

For this project, we would like to develop recognition systems which classify deficiencies in a plant using representation learning which will automatically build features to be used for classification. In the system, there are two components

namely, a recognition subsystem on a server and a mobile application. The recognition subsystem takes a leaf image as an input and the output will be the indication of a nutrient that the plant lacks. The mobile application is used to send input images to the server. It takes a leaf image and filter out the background region before sending it, receives the result back and displays on the screen.

1.2 Objectives and scope of work

The objectives for the project are as follows:

1. To develop plant nutrient deficiency classification system which takes a plant's leaf image as an input.
2. To develop a classification system using representation learning approach.

The scope of the project is listed as follows:

1. The input to the system shall be an image taken in a controlled environment; the background region is dark.
2. The system is to be tested with Vigna Mungo (Black gram).
3. The system shall classify an input leaf image to be deficient in nitrogen, phosphorus, potassium, calcium, iron, magnesium or be complete.
4. The recognition subsystem assumes no background background region.
5. The system assumes only one deficiency presented in the plant.

1.3 Structure of the thesis

The thesis is organized into five chapters The second chapter provides background knowledge needed to fully understand the thesis. In chapter three, methods

that are used to implement the system are explained, it covers the research methods, dataset and environment involved within the system. The experiments will be explained in chapter four. There are seven experiments that was done for parameter configuration , measure selected algorithm and approaches. Chapter five consists of the conclusion and summary part of the thesis.

Chapter 2

Background knowledge

In order to fully understand the explanation of the next chapter we first provide the background knowledge needed. This chapter is separated into two parts. The first part explains about computer vision and image processing in general. The other part describes feed forward neural network and a more specific architectures used in the system.

2.1 Computer vision and image processing

Computer vision is a field which deals with the understanding and interpretation of digital images. In general, it seeks to mimic human ability of analyzing real world information which comes in the form of images in order to extract symbolic information and make decision. Most of the computer vision tasks are concerned with recognition such as object recognition, identification or detection.

Image processing is an operation to transform the representation of images in order to extract more information for human perception or to be more suitable for machine interpretation. *Digital image processing* is used by computers to alter digital images such as image enhancement, image restoration, and image segmentation . Image enhancement is a process to enhance relevant information on a image and

make it more meaningful for a certain application. image restoration is a process to restore damaged image, image segmentation is used to extract or segment a region of interest out from the whole image [13]. Figure 2.1 shows an example of image transformation using techniques in image processing. The left hand side is a color image in RGB color space and another is the image after applied Sobel algorithm which is an edge detection algorithm.

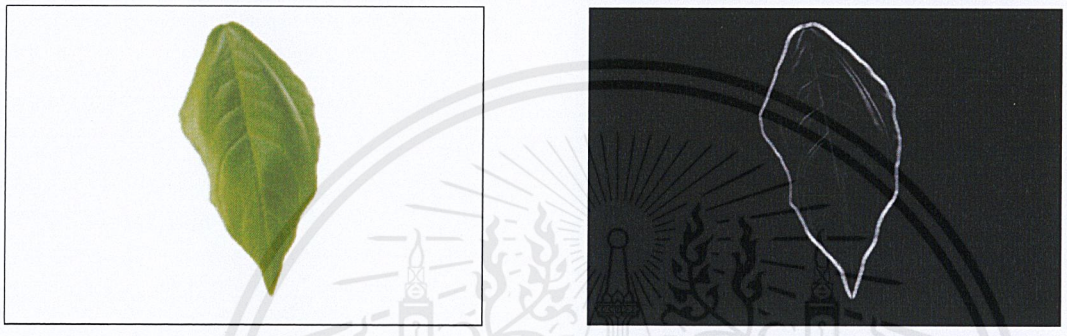


Figure 2.1: Edge detection using Sobel algorithm.

2.2 Active contour model

Image segmentation has a role in computer vision and image processing to extract or annotate the interested region or objects from background. Image segmentation technique used in the system is active contour model or usually called snake (see [8]). Snake uses energy minimization technique where the energy is guided by external constraints and influenced by forces that originated from edges and lines. The goal of active contour model is to find the best approximate to the perimeter of an interested object. The process of active contour model requires some inputs that must be used. These inputs are an input image itself and an initial contour that is drawn by the user or set by the application. Then the snake starts attracting towards the target. As stated earlier that snake is a technique that is based on energy minimization influenced by constraints and image energy. There are two types of energy namely internal and external energy resisting each other. This can be represented as

an energy function as follows:

$$E = \int (\alpha E_{cont} + \beta E_{curv} + \gamma E_{image}) ds, \tag{2.1}$$

where E_{cont}, E_{curv} and E_{image} are energy terms that occur on the contour, where E_{cont}, E_{curv} are energy inside the contour and E_{image} is an external energy. Variable α, β, γ represent relative weights influencing each energy term.

2.2.1 Continuity energy (E_{cont})

It is an internal energy controlling the contour to be continuous. This energy forces the contour to be membrane-like which helps snake to attract to straight lines easier. The energy can be calculated as

$$E_{cont} = ||p_i - p_{i-1}||^2, \tag{2.2}$$

where p_i are points on the contour and $1 \leq i < \infty$. The energy alone minimizes the distance between two points and causes the contour to shrink. The better way to calculate the energy is by subtracting the distance between two points from the mean distance as follows:

$$E_{cont} = (\bar{d} - ||p_i - p_{i-1}||)^2. \tag{2.3}$$

In this way, two points are kept equally apart from each other.

2.2.2 Curvature energy (E_{curv})

Curvature energy controls the smoothness of the contour. The energy eliminates oscillations and keeps the contour smooth. The curvature energy is calculated as

$$E_{curv} = ||p_{i-1} - 2p_i + p_{i+1}||^2. \tag{2.4}$$

2.2.3 Image energy (E_{image})

Image energy is used to attract the contour towards the nearest target boundary. The energy can be calculated from

$$E_{image} = -||\Delta I||^2, \tag{2.5}$$

where the ΔI is a gradient of intensity in each snake point. The reason that Eq. (2.5) is negative is because it is an external force resisting the previous two energies.

2.2.4 Discrete formulation

In order to do a computation of the energy function on a computer, we use summation of estimated points in the contour to approximate the integration.

$$E = \sum_{i=1}^{N} \alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image} \tag{2.6}$$

During the iterative process, snake stops altering a point if it moves less then a threshold. It also stops if the number of iterations exceeds the maximum number of iterations. The procedures of the technique is shown in Algorithm 1.

Algorithm 1 Snake algorithm

MAX_ITERATION = 20

MIN_MOVED_POINTS = 5

current_iteration = 0

count_moved_points = 0

repeat

for each point p_i in a contour **do**

for each point in block size 15×15 which p_i is the center point **do**

Find $E_{continuity}$, Find $E_{curvature}$ and Find E_{image}

end for

Normalize $E_{continuity}$, Normalize $E_{curvature}$ and Normalize E_{image}

min_energy = INT_MAX

new_min_energy_point = p_i

for each point p in block size 15×15 which p_i is the center point **do**

Find $E_{snake}(p)$

if $E_{snake}(p) <$ min_energy OR ($E_{snake}(p) =$ min_energy AND p is the center point) **then**

new_min_energy_point = p

end if

end for

if $p_i \neq$ new_min_energy_point **then**

p_i = new_min_energy_point

count_moved_points = count_moved_points + 1

end if

end for

Update $\beta(s)$'s value

current_iteration = current_iteration + 1

until current_iteration $\leq$ MAX_ITERATION AND count_moved_points $\geq$ MIN_MOVED_POINTS

2.3 Feedforward neural network

Feedforward neural networks, also known as multilayer perceptrons (MLPs), are essential to machine learning nowadays. It is originated in the study of mathematical representations of information processing in biological systems [2]. An MLP composes of neurons or nodes each representing a variable in the network organized in layers. Information flows from the input layer through the network to output layer. Each node output its value to the connected nodes in the next layer with some weight parameters. The first and the last layer contains input and output variables respectively. The intermediate layers are called hidden layers. A single hidden layer MLP is depicted in Fig. 2.2. The variables $\mathbf{X} = [x_1, x_2]$, $\mathbf{A}^l = [a_1^l, a_2^l, ..., a_N^l]$ is a vector of outputs of each node and $\mathbf{O}^l = [o_1^l, o_2^l, ..., o_N^l]$ is a vector of activated output of each node in layer l having N nodes defined as

$$\mathbf{O}^l = f(\mathbf{A}^l), \tag{2.7}$$

where $f(x)$ is a nonlinear activation function which is used to transform the output in a nonlinear fashion which will be explained further shortly. A weight connecting the i^{th} node in layer l to the j^{th} node in layer $l+1$ is represented by an element $w_{i,j}^l$ of weight matrix $\mathbf{W}$.

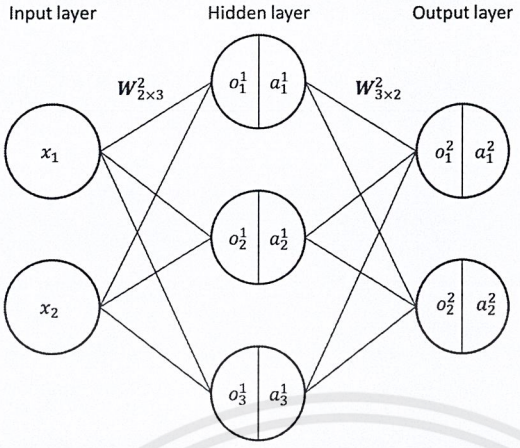


Figure 2.2: Single hidden layer feedforward neural network.

The output value of each node is computed by summing the weighted outputs from the previous layer and is defined by

$$o^l_j = \sum_{i=0}^{M} w^l_{i,j} a^l_i + b^l_j, \tag{2.8}$$

where M is the number of nodes in layer $l-1$ and b^l_j is a constant weight called bias for o^l_j. Note that a^l_i is equal to x_i for the case of $l = 1$ and o^l_i is the prediction for class i if l is the last layer. Once the output is computed, it is then passed to an activation function which has to be chosen from various functions. The most widely used non-linearity today for activation function is rectified linear unit (ReLU) defined in Eq. (2.9) [11].

$$f(x) = max(0,x) \tag{2.9}$$

2.4 Convolutional neural network

Convolutional neural networks (CNNs) are a variant of neural networks which are specifically used for data that comes in a grid-like form such as sequences and images [4]. The term "convolutional" comes from the fact that CNNs perform convolution operation(at least for one layer) instead of normal weighted sum as in MLP

from the last section.

Convolution operation is a way to compute weighted sum in which a set of weights, called a kernel or a filter, is shared between local patches in the input. For 2D discrete data such as digital images, a convolution of input I with a kernel W of size $K \times K$ is defined as

$$\begin{aligned} O(m,n) &= X(m,n) * W(m,n) \\ &= \sum_{i=-\frac{K-1}{2}}^{\frac{K-1}{2}} \sum_{j=-\frac{K-1}{2}}^{\frac{K-1}{2}} W(i,j)X(m-i,n-j). \end{aligned} \tag{2.10}$$

In general, a CNN is organized as stages where the early stages comprises two types of layers namely convolutional and pooling layers [11]. Nodes in a convolutional layer are organized planes called feature maps as in Fig. 2.3. In convolutional layer, inputs are convolved with a set of weights called a filter to produce the output feature map which contains certain features. Different inputs are convolved with different filters. Also, an input is convolved with several filter banks to extract different features. For example, in Fig. 2.3, there are three filter in a filter bank in the first convolutional layer since there are three feature maps in the layer. The resulting feature maps are then passed to a non-linear funciton such as ReLU.

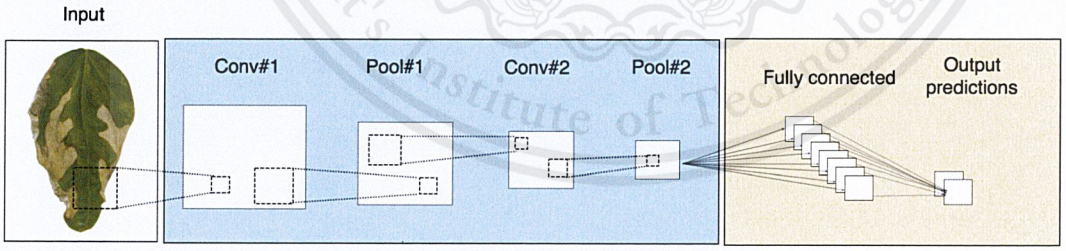


Figure 2.3: Convolutional neural network.

After an activation function is the next layer, pooling layer. In pooling layer, values of small areas in the input are statistically summarized into one unit in the output. Various types of pooling might be used. For example, max pooling outputs the maximum value within the area, average pooling outputs the average of the

values. The stage are repeated two to three times followed by more convolutional layers. The last stage contains several fully connected layers and eventually output layer showing predictions for each class.

2.5 Model training

A neural network is trained using back propagation and gradient descent algorithms. In back propagation, chain rule in derivative is used to calculate gradient of the error function with respect to each parameter and the weight parameters are then adjusted accordingly using gradient descent.

2.5.1 Back propagation

The algorithm progresses by propagating the error or cost back from the output through the network to compute the gradient. The cost function E is usually chosen to be half of the difference between the expected output and the actual output as shown in Eq. (2.11).

$$E = \frac{1}{2} \sum_{i \in out} (t_i - a_i^L)^2, \tag{2.11}$$

where t_i is the label and a_i^L is the actual output prediction for class i. We first give the example of a back propagation in a fully connected layer. The network is shown in Fig. 2.4. The value in the left side of a node is a weighted sum of the nodes in the previous layer and the value in the right side is the output of passing it to ReLU. The objective here is to achieve a partial derivative of the error function with respect to each weight parameter in the network. Using chain rule, the equation for the gradient is as shown in Eq. (2.12).

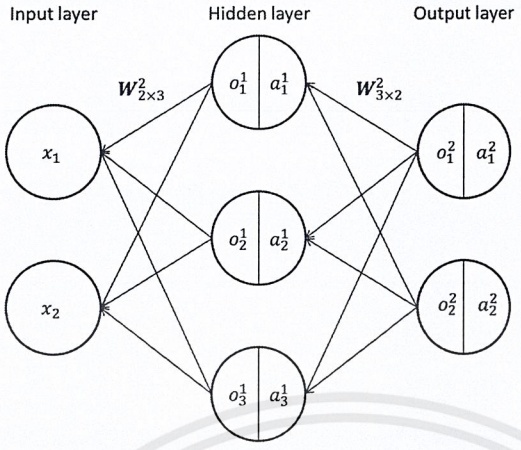


Figure 2.4: An example of back propagation in fully connected neural network In back propagation, error flows from the output layer on the left through the network to the input layer in order to compute the gradient of the error function with respect to each weight parameter

$$\begin{aligned}\frac{\partial E}{\partial w^l_{i,j}} &= \frac{\partial E}{\partial o^l_j}\frac{\partial o^l_j}{\partial w^l_{i,j}} \\ &= \frac{\partial E}{\partial a^l_j}\frac{\partial a^l_j}{\partial o^l_j}\frac{\partial o^l_j}{\partial w^l_{i,j}} \\ &= \frac{\partial E}{\partial a^l_j}\frac{\partial a^l_j}{\partial o^l_j}\frac{\partial o^l_j}{\partial w^l_{i,j}},\end{aligned} \tag{2.12}$$

where $w^l_{i,j}$ is the weight connecting the i^{th} node in layer l to the j^{th} node in layer $l+1$, o^l_j is the value of node j in layer l and a^l_j is the activated version of o^l_j defined as $a^l_j = ReLU(o^l_j) = max(0, o^l_j)$. Each component of Eq. (2.12) can be calculated as follows:

$$\frac{\partial E}{\partial a^l_i} = \begin{cases} \sum_{o_j \in l+1} w^{l+1}_{i,j}\frac{\partial E}{\partial o^{l+1}_j}; l \neq L \\ \frac{\partial \sum_{a_n \in L}(t_n - a^l_n)^2}{a^l_j} = (t_j - a^l_j); l = L \end{cases} \tag{2.13}$$

$$\frac{\partial a_j^l}{\partial o_j^l} = \frac{\partial max(0, o_j^l)}{\partial o_j^l} = \begin{cases} 0; x < 0 \\ 1; x \geqslant 0 \end{cases} \tag{2.14}$$

$$\frac{\partial o_j^l}{\partial w_{i,j}^l} = \frac{\partial \sum_{n \in l-1} a_n^{l-1} w_{n,j}^l + b_j^l}{\partial w_{i,j}^l} = a_i^{l-1} \tag{2.15}$$

Next, we give an example of back propagation in a convolutional layer. The nodes in a convolutional layer is passed to a non-linear function such as rectified linear unit for this case and is typically followed by a pooling layer which is chosen to be max pooling in this case. Only the nodes presented in the pooling layer are used to calculate the gradient of the weight parameters. The weight gradients are calculated as in the following equations:

$$\begin{aligned} \frac{\partial E}{\partial w_{i,j}^l} &= \sum_{i'} \sum_{j'} \frac{\partial E}{\partial o_{i',j'}} \frac{o_{i',j'}}{w_{i,j}^l} \\ &= \sum_{i'} \sum_{j'} \frac{\partial E}{\partial a_{i',j'}} \frac{\partial a_{i',j'}}{\partial o_{i',j'}} \frac{o_{i',j'}}{w_{i,j}}, \end{aligned} \tag{2.16}$$

$$\begin{aligned} \frac{\partial o_{i',j'}}{\partial w_{i,j}^l} &= \frac{\partial \left(\sum_{i''=0}^{K-1} \sum_{j''=0}^{K-1} w_{i'',j''}^l p_{i'-i'',j'-j''}^{l-1} \right)}{\partial w_{i,j}^l} \\ &= p_{i'-i,j'-j}^{l-1}, \end{aligned} \tag{2.17}$$

$$\frac{\partial E}{\partial a_{i,j}} = \sum_{a_{i',j'} \in pool^l} \frac{\partial E}{\partial o_{i',j'}^{l+1}} \frac{\partial o_{i',j'}^{l+1}}{\partial a_{i',j'}^l}, \tag{2.18}$$

$$\begin{aligned}\frac{\partial o_{i',j'}^{l+1}}{\partial a_{i,j}^{l}} &= \frac{\partial}{\partial a_{i,j}^{l}}\left(\sum_{i''}\sum_{j''} w_{i'',j''}^{l+1} a_{i'-i'',j'-j''}^{l} + \mathbf{b}^{l+1}\right) \\ &= w_{i'-i,j'-j}^{l+1},\end{aligned} \tag{2.19}$$

where $\mathbf{W}^l$ is a filter a in convolutional layer l, $\mathbf{O}^l$ is a feature map after convolution. Substitute Eq. (2.19) in Eq. (2.18), we get Eq. (2.20).

$$\frac{\partial E}{\partial a_{i,j}} = \sum_{a_{i',j'} \in pool^l} \frac{\partial E}{\partial o_{i',j'}^{l+1}} w_{i'-i,j'-j}^{l+1}, \tag{2.20}$$

2.5.2 Gradient descent

Once the gradient of the error function with respect to each weight parameter is computed, the weights are adjusted using gradient descent as in the following equation.

$$w_{i,j}^{l} = w_{i,j}^{l} - \alpha \frac{\partial E}{\partial w_{i,j}^{l}} \tag{2.21}$$

where α is a predefined learning rate. All the weight parameters are adjusted iteratively until convergence.

2.6 Convolutional neural network variants

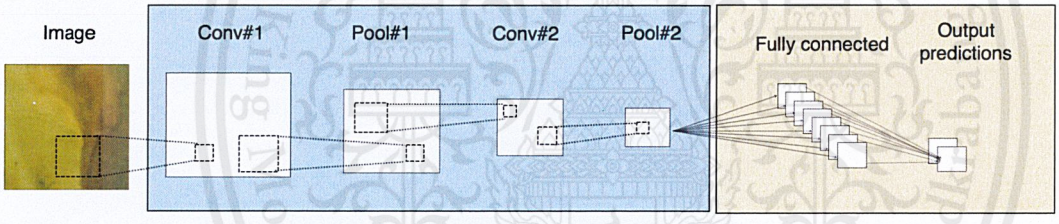
There exist numerous variants of convolutional neural network models with different specialization. Some of the models widely known for good performance on object recognition tasks are LeNet and AlexNet which are covered in this section.

2.6.1 LeNet

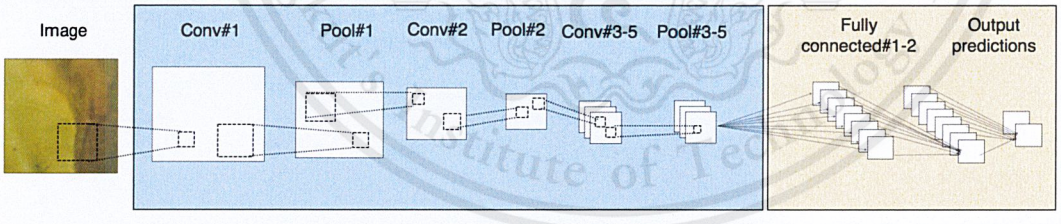
LeNet was introduced as a convolutional neural network model specialized in classifying handwritten characters [11]. It comprises two alternation of convolutional and max-pooling layers followed by two fully connected layers as shown in Fig. 2.5a. The model uses a sigmoid function as an activation function for each layer accept the output layer which uses a Radial Basis Function (RBF). The output response y_i for each class i is defined as

$$y_i = \sum_j (x_j - w_{i,j})^2, \tag{2.22}$$

where x_j is an output of a node j in the layer before output and $w_{i,j}$ is a weight parameter connecting the two nodes i and j.



(a) LeNet



(b) AlexNet

Figure 2.5: Overall architecture of LeNet and AlexNet models (a) LeNet composes of two alternation of convolutional and pooling layers followed by two fully connected layers. (b) AlexNet comprises two alternation of convolutional and pooling layers followed by two convolutional, a pooling, and three fully connected layers.

2.6.2 AlexNet

AlexNet is widely known for outstanding performance in ILSVRC (ImageNet Large Scale Visual Recognition Competition) in 2010 [9]. In general, the network consists of three alteration of convolutional and max-pooling layers followed by two convolutional and three fully connected layers as shown in Fig. 2.5b. The unique features of the model is briefly presented as follows:

- ReLU – An activation function used is ReLU.
- Local response normalization – Normalization scheme used for better generalization. It is applied to each activated output of each node after the first two pooling layers. For each activated output $a^L_{i,j}$, its normalized response $b^L_{i,j}$ is defined as

$$b^L_{i,j} = a^L_{i,j} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} \left(^L_{i,j}\right)^2 \right)^{\beta}, \tag{2.23}$$

  where n is the number of adjacent filters used to normalize $a^L_{i,j}$, N is the number of total filters in the layer, and k, α, β are predefined parameters.
- Overlapping pooling – For pooling layers, each position of a pooling window overlaps.

The model also employs a overfitting prevention technique used called "drop out". When training with dropout, each hidden node has a chance, with probability 0.5, to be assigned with zero as output in a forward pass.This lets the output created by the network in each forward pass be produced with different architectures.

2.7 Extreme learning machine

Extreme learning machine (ELM) , proposed by [6], is a learning procedure for a single hidden-layer feed forward neural network (SLFN). Instead of using iterative

algorithm such as back propagation to learn the weights of the network, it uses an analytical method to compute them by formulating the network computation as a matrix equation with zero error and computing the weights by using matrix inverse.

In general, feed forward neural network with traditional learning procedure consumes a lot of time due to the gradient-based cost minimization methods. Also, the parameters in the network need to be iteratively adjusted. ELM overcomes the difficulties in iterative learning methods by randomly generating the input weights and the hidden layer biases and analytically calculate the output weights using matrix inverse. A SLFN is shown on Fig. 2.6 where we have $\mathbf{x} = [x_1, x_2, x_3]$ as an input feature vector to the network, $z_i = g(\mathbf{x}\mathbf{W}_i^T)$ is the activation function of the hidden layer neurons, b_1, b_2, b_3 are hidden layer biases,

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}, \tag{2.24}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \beta_{31} & \beta_{32} & \beta_{33} \end{bmatrix}, \tag{2.25}$$

are weights to be applied to the input vector and output from the hidden layer respectively.

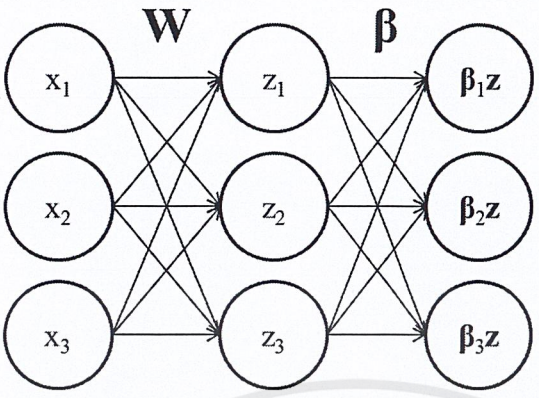


Figure 2.6: Single hidden-layer feed forward neural network. The left most layer is the input layer where a feature vector is fed to the network. The middle layer is the hidden layer where the feature vector is multiplied by the weights of the hidden layer and added by the bias terms

With N samples (x_i, t_i) where $x_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^T$, $\mathbf{t}_i = [t_i 1, t_i 2, \cdots, t_{im}]^T$ and $i = 0, 1, 2, \cdots, N$, hence the network with n input and m output neurons, an SLFN with M hidden neurons then could be mathematically described as the following equation for each jth training sample:

$$\sum_{i=1}^{M} g(\mathbf{w}_i \mathbf{x}_j^T + b_i)\boldsymbol{\beta}_i = \mathbf{o}_j, \tag{2.26}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$ is the weight vector from the input neurons to the ith hidden neuron and $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is the weight vector from the ith hidden neuron to the output neurons b_i is the bias term of the ith hidden neuron and $\mathbf{o}_j = [o_1, o_2, \cdots, o_N]$ is the actual output for the jth sample. To have zero error meaning there are values of β_i, b_i, w_i satisfying

$$\sum_{i=1}^{M} g(\mathbf{w}_i \mathbf{x}_j^T + b_i)\boldsymbol{\beta}_i = \mathbf{t}_j, \tag{2.27}$$

where $\mathbf{t}_j = [t_1, t_2, \cdots, t_N]$ is the label for jth sample. Eq. (2.27) can be written as a matrix equation as $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, where

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N \times m}, \tag{2.28}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_N^T \end{bmatrix}_{M \times m}, \tag{2.29}$$

$$\boldsymbol{H} = \begin{bmatrix} g(\boldsymbol{w}_1 \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_M \boldsymbol{x}_1 + b_M) \\ \vdots & \vdots & \vdots \\ g(\boldsymbol{w}_1 \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_M \boldsymbol{x}_N + b_M) \end{bmatrix}_{M \times m} \tag{2.30}$$

The matrix H is called the hidden layer output matrix of the neural network. ELM simply random initial values for matrix **W** so that matrix **H** can be calculated. Once **H** is known, the learning procedure seeks to find matrix $\boldsymbol{\beta}$ by solving equation

$$\boldsymbol{\beta} = \mathbf{H}^{+}\mathbf{T}, \tag{2.31}$$

where $\mathbf{H}^{+}$ is a Moore–Penrose generalized inverse of **H**. **W** and $\boldsymbol{\beta}$ are then used as weights that produce zero error for the whole training data.

Chapter 3

Methodology

In general, the system functions by letting the user use a mobile application to take a photo of a leaf of interest. The application then sends the image to a server to be processed. The server then classifies the deficiency presented in the input image and send the result back to the mobile application. The overall process is depicted in Fig. 3.1

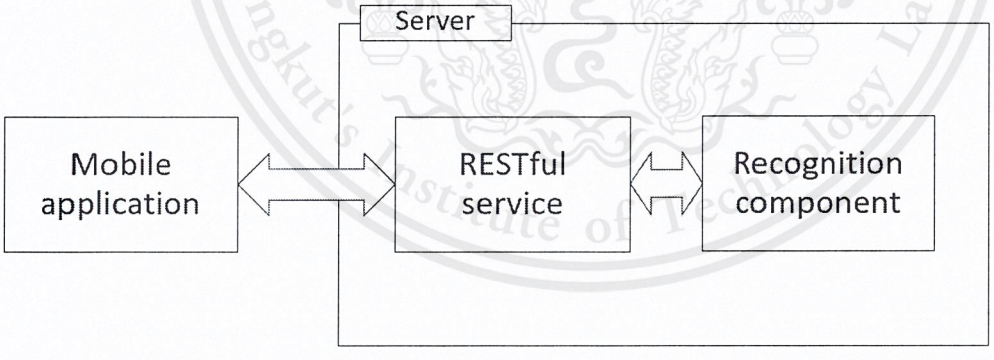


Figure 3.1: Overview of the system processes.

Detailed methodologies applied in developing each component of the proposed system is explained in this chapter.The chapter is subdivided into three parts which are mobile application component, server component and recognition component.

3.1 Mobile application component

A mobile application called *DeficiencyDetector* is responsible for letting the user take a leaf image and send it to the server to be processed by recognition component. Figure 3.2 shows the first page of the mobile application, which is used to acquire a leaf image from the user. After an image is obtained, the application proceeds to the image segmentation process.

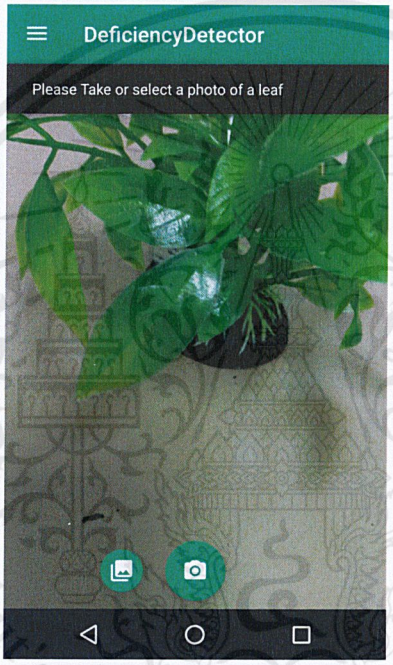


Figure 3.2: The first page of the mobile application.

The user must draw a contour around the leaf of interest in order to segment it from the background. Figure 3.4 shows an example of initial contour drawn by the user. The segmentation technique used in this application is active contour model which is often called snake. Figure 3.3 shows the result of segmentation process.

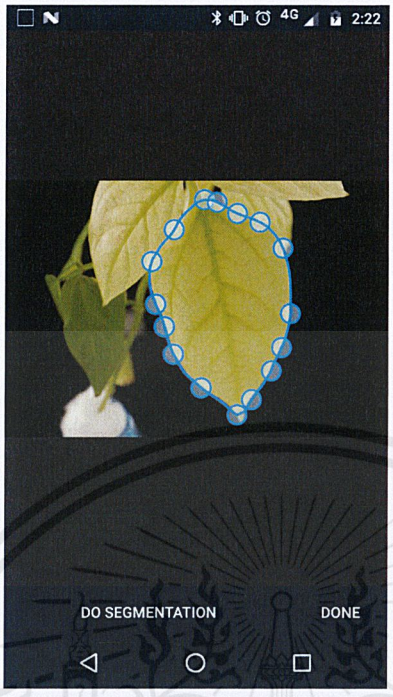


Figure 3.3: The example of result in segmentation page of the mobile application.

After the segmentation process is completed, the mobile application shows the result of the segmentation and allows the user to adjust the points around the boundary of the interested leaf in order to help the segmentation algorithm in case the drawn contour does not match with the actual leaf. Figures 3.5 and 3.6 show examples of the result in each iteration of adjusting the contour. The user can decide to run the segmentation again or send the segmented image to the API service which is a RESTful service. RESTful service is a way to provide information or data through the Internet. This allows a client to access information by using uniform or predefined request methods or usually called "HTTP verbs". Normally, REST responses with data format such as JSON, HTML or XML back to the client after receiving a request.

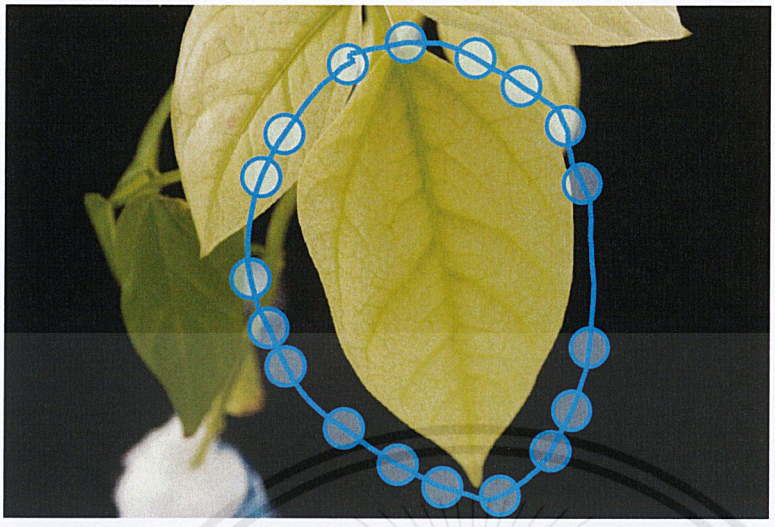
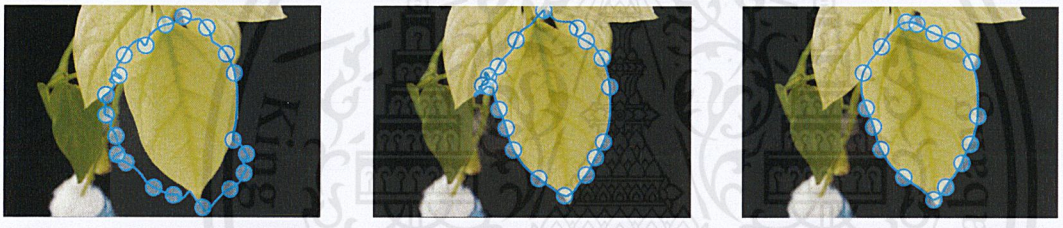


Figure 3.4: Example of an initial contour drawn by the user.



(a) 1st time segmentation (b) 2nd time segmentation (c) 3rd time segmentation

Figure 3.5: Example of interactive active contour model segmentation.

Figure 3.6: Example of the result to be sent to the server.

3.2 Server component

This component is responsible for handling application requests. Server component or API service allows the application to upload a background segmented image to be processed and returns the result using a RESTful service. The method path that API allows mobile application to upload image is

`recognition/upload/image`,

where the method and input required are POST and segmented images respectively. After the application uploads an image to the server, the server passes the image to the recognition component. The process is shown in Fig. 3.7.

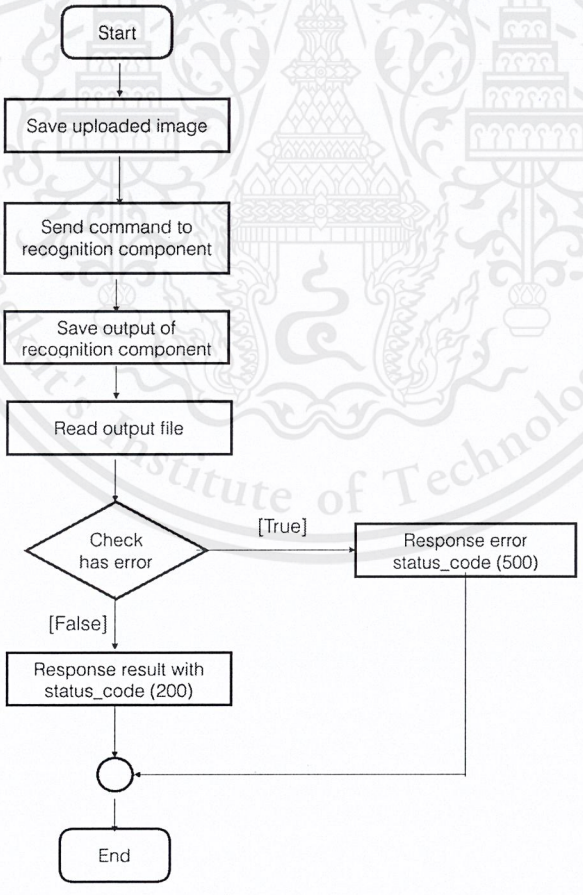


Figure 3.7: Flowchart of overall operations after an image is uploaded.

In Fig. 3.7, the server saves the uploaded image and sends it to the recognition component by using command line code as follows:

```
"executable file in .exe" "arguments" > "outputfile in .txt"
```

The command line code lets the system run the executable file and records the output to a text file. After that, the server reads the output file and check for errors. If there is an error, the status code that response to user will be 500 (Internal Server Error). If no error occurs, this component responses the result and status code 200 (OK).

3.3 Recognition component

Recognition component is responsible for the classification of leaf images fetched from the mobile application. Overall, there are three steps as follows:

- Block extraction — An input image is divided into small blocks. They are then fed to block decision maker.
- Block decision making — All blocks are labeled with their corresponding deficiency classes.
- Leaf decision making — All block labels are used to make the final decision regarding the deficiency of the whole leaf image.

The overall process is shown in Fig. 3.8.

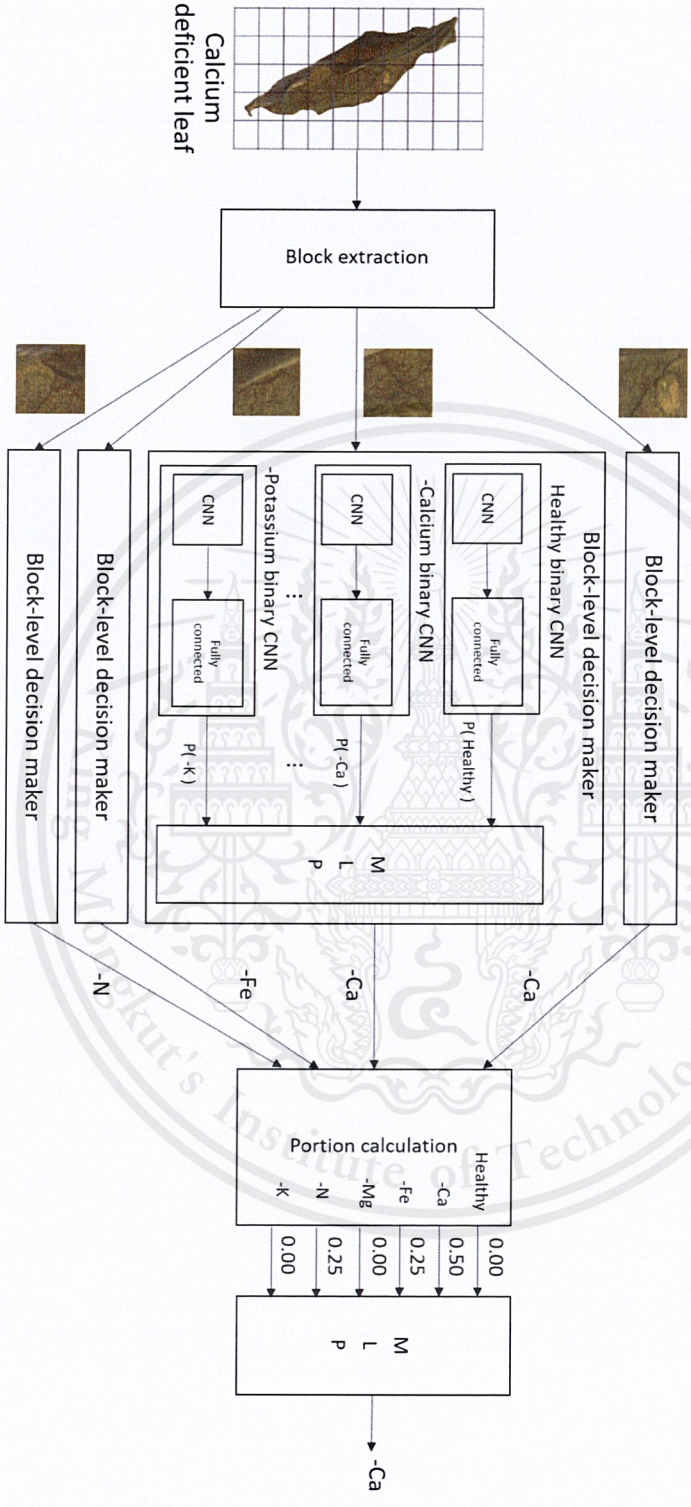


Figure 3.8: Flowchart showing overall process of recognition component.

3.3.1 Block extraction

Each input leaf image is first divided into blocks of size 64×64 pixels. The extracted blocks contain no background region. The reason for using blocks instead of the whole leaf image is that the number of leaf images used for training is not enough to cope with the variety of leaf shapes and sizes. Using blocks instead of the whole leaf image helps the classification to be less dependent of leaf shapes and sizes.Fig. 3.9 shows an input leaf image divided into blocks and blocks that are qualified to be fed to the next process (highlighted blocks).

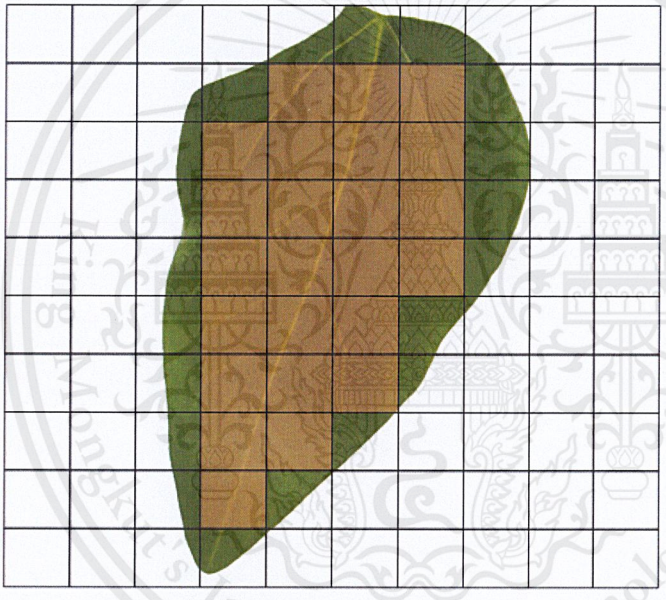


Figure 3.9: Pre-processing is done by subdividing an input image into blocks of size 64×64 pixels containing only leaf regions. Only the highlighted blocks are fetched to the network.

3.3.2 Block-level decision making

After the input image is divided into blocks, the blocks are then fed to a one-vs-all classifier to determine the deficiency presented in the block. The overall process is depicted in Fig. 3.10

One-vs-all classifier

In one-vs-all classifier, a block is fed to a set of binary CNN. A binary CNN is a convolutional neural network that only focuses on one deficiency class which will be explained in Section 3.3.2. There are in total of six binary CNNs corresponding to each deficiency class. Each of these binary CNN will yield a probability that the input block belongs to the corresponding deficiency class. After that, the block will be associated with six probabilities indicating how probable the input block is deficient in each class. These probabilities are then used as features of the block and are fed to a multilayer perceptron (MLP) trained with ELM to make a final decision about the deficiency of the block.

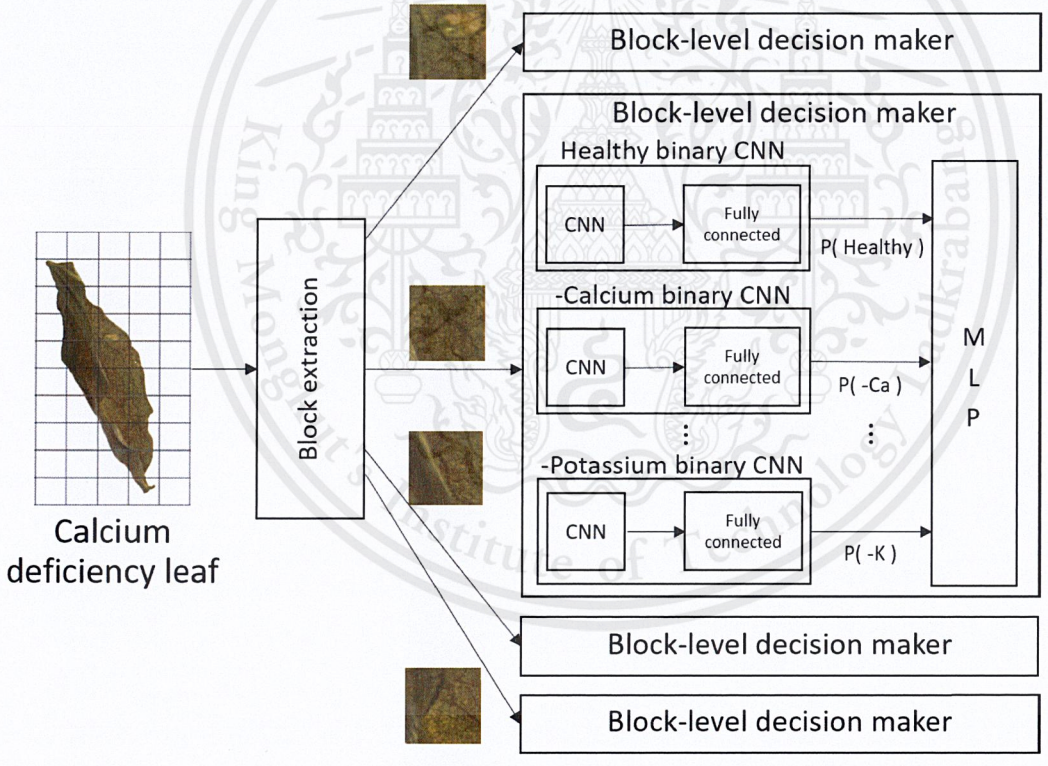


Figure 3.10: Overall process of block decision making which consists of a set of six convolutional neural networks and an extreme learning machine

Binary convolutional neural network

Binary CNN classifier is a convolutional neural network which only determines that the input block belongs to the interested class or not. It is implemented using Caffe framework [7]. However, hyperparameters of the network needs to be manually configured.

Common strategy of designing a network architecture is to start with the existing well-known architecture and tune them to be suitable for the data. There are numerous existing CNN architecture for image recognition task. Some of them are well known in object classification such as GoogleNet, Microsoft ResNet, AlexNet. All these architectures win ImageNet competition which is an object classification competition using images from huge dataset with 1.4 million high resolution images over 1,000 categories. The convolutional neural network used in the proposed system was initially based on AlexNet. The reason to use it is because the architecture performs well in object classification and its size is smaller comparing to the other networks. However, it still needs modification and configuration for making it perform well for the system. Fig. 3.11 shows the system's network architecture, it consists of three convolutional layers with filter size of 11, with no padding and stride. The second layer uses a filter of size 5, 2 pixels padding and no stride. The last convolutional layer has the same parameter as the previous layer. There are also two pooling layer after the first two convolutional layers with the same parameter setting which is filter of size 3 and padding 2 pixels. The last three layers are fully connected layers. The first two each consists of 2,048 neurons. and the last layer is an output layer with two neurons.

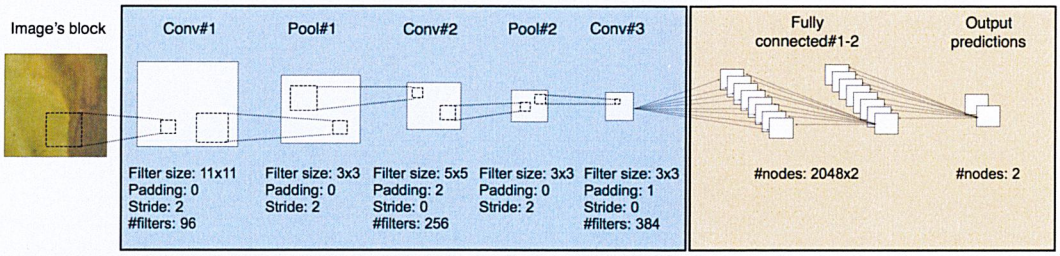
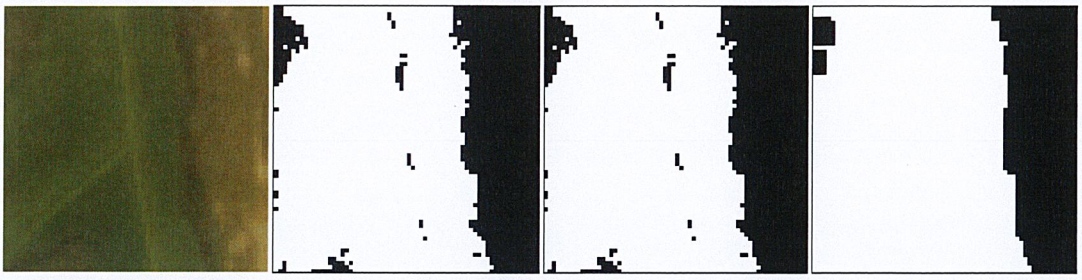


Figure 3.11: Network architecture and layer's parameter for every binary CNN.

Network training

After the architecture and hyperparameters of the neural network are properly configured, the network then needs to be trained with the prepared data set in order to learn the features of deficient leaves in each nutrient. Most of the existing neural network models initialize weight parameters with Gaussian or uniform distribution. The network is trained in a supervised fashion using back propagation as explained in section 2.5. The training is done by feeding input images with a label indicating the expected output from the network. The network uses these label information to adjust weights.

The whole dataset is separated into three categories which are training set, validation set and testing set. Training set contains samples used for actually tuning the network parameters. Validation set is used to measure generalization performance during training process. Testing set is used for measuring accuracy for the trained network. Fig. 3.14 describes process flow for network training. Before feeding an input leaf image into the network, a preprocessing step is required. The image is first divided into blocks as in Section 3.3.1. Each block is then analyzed and only the blocks with deficiency symptoms are used to train each binary CNN (except for the one that classifies healthy blocks).



(a) Original block (b) After Mapping (c) After thresholding (d) After opening

Figure 3.12: Example of a deficient block in each process of the histogram analysis.

Fig. 3.12 shows an example of an image being in each stage of the histogram analysis. To analyze each block, a histogram analysis method is used. A histogram is formed from hue channel of 100 healthy leaves' block images taken in the first day. It is used to model the distribution of healthy leaf's color. The number of bin in the histogram is set to be 26. The hue value of each block image being processed is then mapped with the value in the corresponding bin in the histogram as in Fig. 3.12c. Histogram bins having relatively low value should be considered as noises and eliminated. Hence, the pixels which fall below a threshold value, meaning it is quite different from the model, are set to zeros (absolute black). The threshold T is calculated as follows:

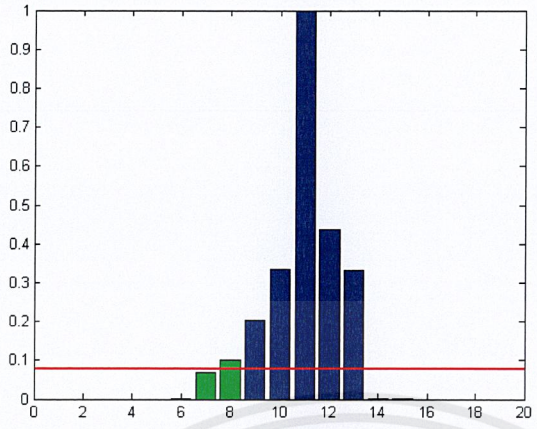
$$T = \frac{h_i + h_{i+1}}{2} \tag{3.1}$$

where h_i is the number of samples in the last bin i having value less than 5% of total values in the histogram. To give an example for this, suppose that a histogram in Fig. 3.13, having total values from all bins equal 2.4792, is obtained from the training images. A threshold value is calculated by first constructing a cumulative histogram. Then , in the cumulative histogram, let i^{th} bin be the last bin having value less than 5% of all the last cumulative bin which equals $0.05 \times 2.4792 = 0.124$. The i^{th} cumulative bin in this case is the red bin in Fig. 3.13b which has a value of

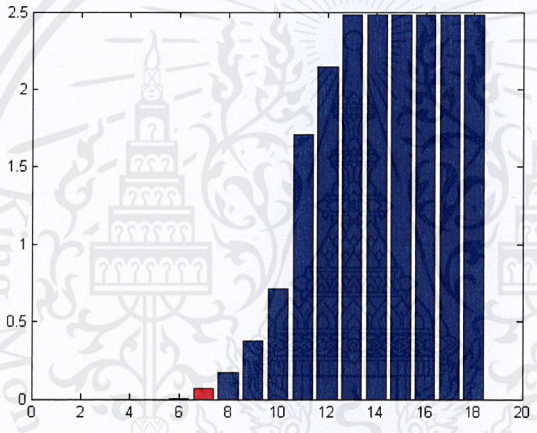
0.0701. Threshold T is then calculated using h_i and h_{i+1} from the corresponding i^{th} and $(i+1)^{th}$ bins, which are the two green bins in Fig. 3.13a, of the normal histogram as in Eq. (3.1). Therefore, we have $T = \frac{0.0701+0.101}{2} = 0.0856$ which is a red line in Fig. 3.13a.

Following, opening morphological operation with a rectangular structure of size 5×5 is applied to the thresholded image.The effect of the process can be seen in Fig. 3.12d. If the portion of zeros in the resulting block image is less than 10%, the block is considered normal. Normal blocks will be discarded and will not be used for training. For example, a portion of black pixels in an image in Fig. 3.12d is 27%. Therefore, it is considered a deficient block and will be used for training.

After that, the system feeds each preprocessed image to the neural network. Then the network compute the values for each output node based on the input images. The process is called forward pass. Gradient values are computed from the output nodes. Then the weights are adjusted based on these gradient values. The entire process repeats until all images are fed into the network.



(a) Original histogram



(b) Accumulative histogram

Figure 3.13: Example of a histogram used in histogram analysis.

Block labeling

For each block, the probabilities acquired from the six binary CNNs are fed into an MLP to determine the block's final deficiency class. Each block is labeled is labeled with the result from the MLP. The MLP has six input nodes for each probability, seven hidden nodes, and six output nodes for each resulting prediction of deficiency class.

The blocks used to train each binary CNN are passed into each binary CNN

to obtain their six probability values. The MLP is trained using these probability values.

3.3.3 Leaf-level decision making

Once each block in a leaf is labeled with a deficiency class, the whole leaf can now be classified. To be invariant from number of blocks in a leaf, the features used to classify a whole leaf are portions of labels on each block in the leaf. The features of each leaf are fed into an MLP with six hidden nodes and six output nodes for each deficiency class to be predicted for the leaf. The result from the MLP is the final prediction of the deficiency class of the input leaf and will be send back to the mobile application

The MLP is trained with ELM using labels' portions of each leaf formed by training blocks from Section 3.3.2.

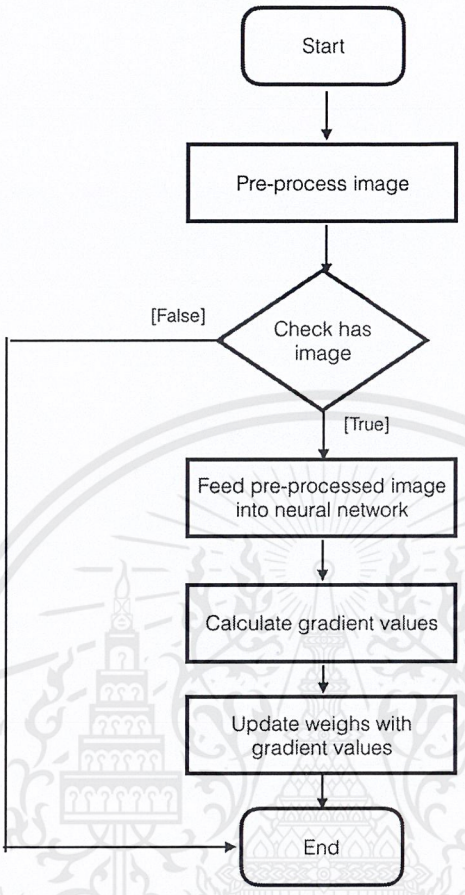


Figure 3.14: Flowchart of network training process of the CNNs.

Chapter 4

Experiments

Many experiments have been conducted to evaluate the methods proposed in the previous chapter and to select the most appropriate configurations for the system in a reasonable way. This chapter describes in details the objectives and processes of each experiment.

4.1 Dataset

The dataset from Computer Vision laboratory (see appendix A), International College, King Mongkut's Institute of Technology Ladkrabang, has been used in the experiments. In the dataset, there are 3,463 leaf images taken using DSLR camera and categorized into six classes which are nutrient deficiency in calcium (-Ca), iron (-Fe), magnesium (-Mg), potassium (-K), nitrogen (-N), and healthy. Each class of leaf images consists of 10 plants observed in 28 days. The distribution of the leaf images of each plant for each class is shown in Table 4.1 .The resolution of the images is 1296 $\times$ 864 pixels. Each image contains a leaf in the center. After all images were taken, the background region in each image was segmented out and replaced with black color using photo editor software. The example of the background segmentation method is depicted in Fig. 4.1. Table 4.2 shows the number of

images in the dataset. The dataset is separated into two types, old leaves and young leaves which are leaves from the lower and upper part of the a plant, respectively.

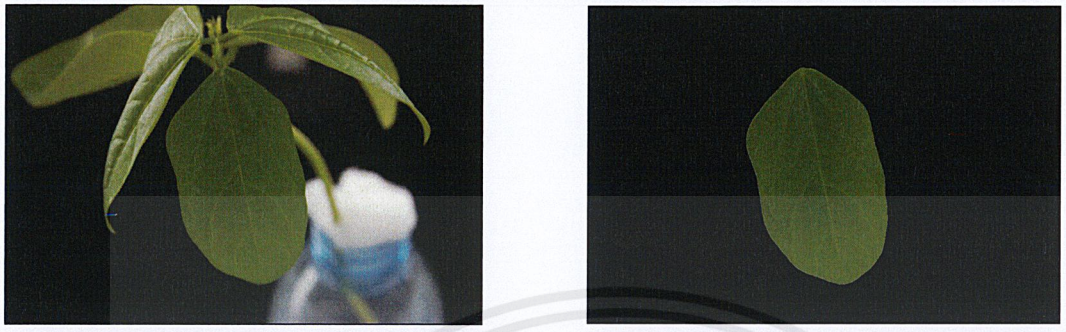


Figure 4.1: The original image (left) before background segmentation and the same image (right) after background segmentation

Table 4.1: The distribution of leaf images from each plant for each deficiency class

Class name	Plant number											
	1	2	3	4	5	6	7	8	9	10	11	12
Healthy	54	52	54	52	50	50	54	54	52	47	53	53
Calcium (-Ca)	40	24	22	40	40	18	40	40	50	51	47	54
Iron (Fe)	54	49	54	51	48	54	28	52	52	47	54	48
Magnesium (-Mg)	46	39	52	54	50	53	54	53	41	52	45	54
Nitrogen (-N)	54	53	53	46	54	44	54	50	50	39	50	47
Potassium (-K)	54	54	54	54	54	54	45	46	50	52	54	54
Phosphorus (-P)	54	53	52	53	53	34	51	53	48	46	49	48

Table 4.2: The number of leaf images in the dataset.

Nutrient deficiency	Number of images		
	Old leaf	Young leaf	Total
Healthy	305	320	625
Calcium (-Ca)	230	236	466
Iron (-Fe)	291	300	591
Magnesium (-Mg)	300	293	593
Nitrogen (-N)	290	304	594
Potassium (-K)	309	285	594
Total	**1,725**	**1,738**	**3,463**

4.2 Measure

The experiments use the following measures as indicators of the classification performance.

- Accuracy – used as the overall indicator of the performance but it does not give much information of the classifiation
- Precision – used to measure how precise the prediction is. In other words, how reliable the output prediction is
- Recall – used to measure how much can the classification correctly predicts a certain class
- F-measure – combines precision and recall for easier evaluation

The measures are calculated as follows:

$$accuracy = \frac{t_p}{N}, \tag{4.1}$$

$$precision = \frac{t_p}{t_p + f_p}, \tag{4.2}$$

$$recall = \frac{t_p}{t_p + f_n}, \tag{4.3}$$

where N is the number of all test samples, t_p, f_p, f_n are true positive, false positive and false negative results respectively.

4.3 Experiments

4.3.1 Experiment 1: Six-output CNN classification

Objective

The experiment was conducted to measure the performance of a multi-class CNN classifier. The overall process of the experiment is depicted in Fig. 4.2.

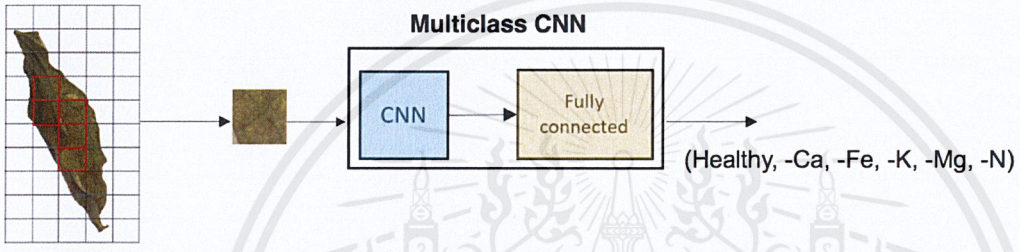


Figure 4.2: The flow of process in six-output CNN classification

Experiment Setup

A CNN is used to classify input leaf image. The outputs from the CNN are the probability of each deficiency class. The class which has the highest probability is assigned to the input image. The first step is to do preprocessing on each input image. Then resulting preprocessed data are fed into the network for classification and the result is evaluated. The parameters of the network and preprocessing are explained in the rest of this section.

Preprocessing: Before feeding the dataset into the network, it requires some preprocessing. This process divides a leaf image into small blocks which contain only leaf area without black background. The size of each block is 64×64 pixels as the shaded grid in Fig. 4.4. An example of a block is shown in Fig. 4.3. Table 4.3 states the number of blocks for each nutrient deficiency. All of the following experiments use these blocks as inputs.

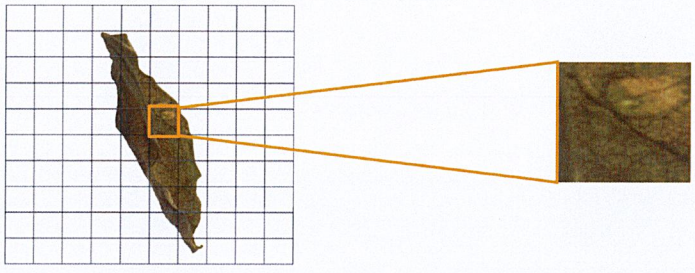


Figure 4.3: The original image (left) before applying preprocessing and a cropped image (right).

Table 4.3: The dataset after cropped images.

Nutrient Deficiency	Quantity (blocks)
Healthy	11,110
Calcium (-Ca)	5,321
Iron (-Fe)	11,094
Magnesium (-Mg)	9,995
Nitrogen (-N)	11,464
Potassium (-K)	10,550
Total	**59,534**

Convolutional neural network architecture and parameters: Figure 4.4 shows network architecture and definition of each layer which is used in this experiment. This network is similar to AlexNet's architecture. Only the first layer's filter size is different since AlexNet uses an input image of size 256×256 and the input size here is 64×64.

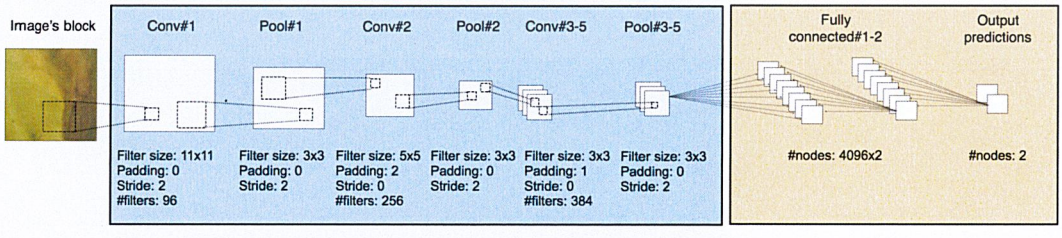


Figure 4.4: Network architecture and parameters of the CNN used in this experiment

Parameter setting: Table 4.4 shows several parameters governing the training process of the network. Batch size is the number of images used to train the network during an iteration. Learning rate is a hyperparameter governing the learning capability of a neural network. Learning policy is a parameter to reduce learning rate as number of iterations increases. The policy helps for making the network running to optimal solution. Weight decay is a parameter used as penalty for large value of weight. In this experiment, the first plant to the sixth plant are selected as a training set, the seventh plant as a validation set, and the eighth to the twelfth as a testing set for evaluating the experiment.

Table 4.4: The learning parameter setting for the network

Parameter name	Value
Batch size (blocks)	64
Snapshot period (iterations)	250
Learning policy (type)	inverse decay
Learning rate α	0.01
Number of iteration (iterations)	55,000
Weight decay	0.0001

Training data distribution: The dataset can be divided into six classes. Table 4.5 shows the number of preprocessed images that are used to train and test the network.

Table 4.5: The number of blocks separated into training set, validating set and testing set

Nutrient Deficiency	Number of blocks		
	Train	Validate	Test
Healthy	5,268	933	4,909
Calcium (-Ca)	1,525	434	3,362
Iron (-Fe)	5,470	415	5,209
Magnesium (-Mg)	4,713	972	4,310
Nitrogen (-N)	5,383	1,244	4,837
Potassium (-K)	5,228	973	4,349
Total	**27,587**	**4,971**	**26,976**

Result and discussion

Averaged accuracies of the classifier on each class for the test set are shown in Table 4.6. The result shows equally low classification accuracy for all the classes except calcium which achieves a significantly low percentage of correctness. This experiment also leads to another experiment about a new approach for improving system performance. Other classes that achieve relatively low accuracy might be caused by

- The portion of the data in each class is not divided equally because some symptoms cause leaves to shrink and leads to low number of extracted blocks.
- Images in the dataset were captured in different camera settings which makes it more difficult for the classifier to learn.
- For all classes of deficiency, the images taken in the early days are quite similar to each other which leads to difficulty in learning.

Table 4.6: The classifier results in average percentage for each nutrient deficiency

Nutrient Deficiency	Accuracy (%)
Healthy	21.86
Calcium (-Ca)	7.10
Iron (-Fe)	25.48
Magnesium (-Mg)	32.63
Nitrogen (-N)	30.28
Potassium (-K)	26.50
Average	**23.98**

4.3.2 Experiment 2: Binary CNN classification

Objective

A new approach is created to solve a problem in the previous experiment. This time, multiple binary CNNs are used instead of one multi-class CNNs. The objective is also to evaluate this new approach. The overall process of the experiment is depicted in Fig. 4.5.

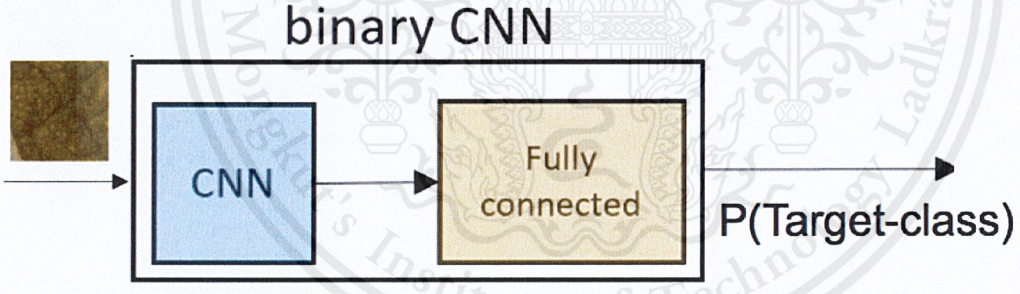


Figure 4.5: The flow of process in binary CNN classifier

Experiment setup

In this experiment, six binary CNNs corresponding to the six deficiency classes are trained. Each classifier only detects the probability that only one certain nutrient is presented in the input image. The input to the network is the same as the network in the previous experiment. The experiment uses the preprocessed dataset from the experiment one see (Section 4.3.1). Details of the configuration are explained next.

Table 4.7: The learning parameters are selected setting for every classifiers.

Parameter name	Value
Batch size (blocks)	64
Create snapshot (iterations)	250
Learning policy (type)	inverse decay
Learning rate α	0.01
Number of iteration (iterations)	20,000
Weight decay	0.0001

Convolutional neural network architecture and parameters: The network used in this experiment is shown in Fig. 4.6. It consists of three convolutional and two pooling layers. The number of layers are reduced because the old one was optimized for more complex data, images of size 256×256 pixels, but the system uses a block with size 64×64. With too much layers and small input size, very deep layers gain no new information and are not able to form new abstract features. This network also reduces the number of neurons in fully connected layers by half because the number of weights connected with the first fully connected layer are reduced. The output of the determines whether a block is deficient in a target class or not.

Parameter setting: All binary CNN use the share the same architecture and parameters. The learning parameters of each CNN is shown in table 4.7.

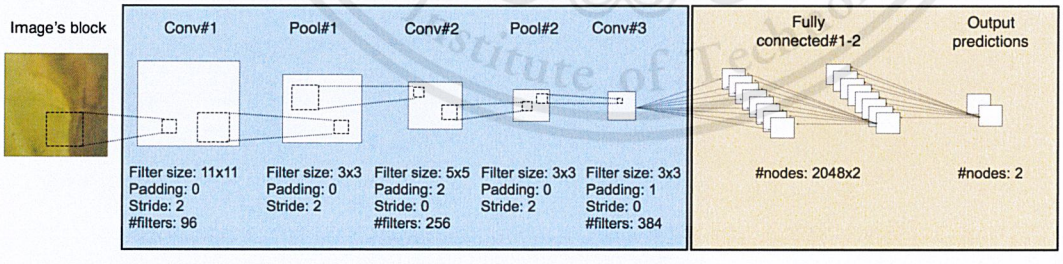


Figure 4.6: The flow of process and network parameter which are used in this experiment

Training data distribution: The dataset can be divided into six classes. The number of extracted blocks for each class used to train and test the network is shown

in Table 4.8. For each binary CNN, each block is only labeled with two classes which are the interested class labeling as zero and others class labeling as one.

Table 4.8: The distribution of data for each binary CNN

Binary CNN	Training data (blocks)		Validation data (blocks)	Testing data (blocks)
	Interested class	Others class		
Healthy	5,268	5,018	933	4,909
Calcium (-Ca)	1,525	1,616	434	3,362
Iron (-Fe)	5,470	5,020	415	5,209
Magnesium (-Mg)	4,713	4,326	972	4,310
Nitrogen (-N)	5,383	5,018	1,244	4,937
Potassium (-K)	5,228	5,020	973	4,349
Total	**27,587**	**26,018**	**4,971**	**26,976**

Result and discussion

In Table 4.9, accuracies on the test set of each CNN and their corresponding number of iterations used to train them are shown. The main difference between this experiment and previous experiment is the number of outputs of the CNNs. In this experiment, multiple binary CNNs are used to perform classification where the last experiment uses only one six-classes CNN. This leads to extreme improvement in accuracies. In other words, each binary CNN only answers one out of two classes in contrast to experiment one where the CNN needs to answer one out of six classes. Therefore, the accuracies in this experiment cannot be compared with those in the previous experiment. Since what obtained in from these binary CNNs are six probabilities of each deficiency class, it is required to make the final decision regarding the actual deficiency presented in the block using these six probabilities. Considering only the performance of the binary CNNs shown in Table 4.9, overall, the accuracies are quite good. The best binary CNN in this case is the one detecting iron deficiency with a relatively high accuracy of around 80%. The rest of them perform quite equally well with around 70% of accuracy except for magnesium classifier which achieves the lowest accuracy of around 60%.

Some binary CNNs achieve relatively low accuracy comparing to others. The issue might be caused by:

- Some symptoms do not spread through the entire leaf which leads to incorrect learning.
- Images in the dataset were captured in different camera parameters which also leads to inappropriate learning.

There is also another problem with the performance of the network.Fig. 4.7 shows a strange behavior of one of the binary CNN. The graph shows the portion of true negative (blue line) and true positive (green line) samples obtained from the classification. The mirroring behavior might be caused from the fact that some deficient blocks are labeled as deficient. However, they do not represent any symptom because symptom might not occur through the whole leaf. To solve the problem, only blocks having abnormal region should be used to train the network except for the one that classifies healthy class which will use the healthy blocks in the dataset as usual.

Table 4.9: The parameters are selected setting for each classifier

Binary CNN	Accuracy (%)	Number of iterations
Healthy	70.32	250
Calcium (-Ca)	72.72	13,500
Iron (-Fe)	80.05	2,500
Magnesium (-Mg)	62.16	10,500
Nitrogen (-N)	73.20	500
Potassium (-K)	68.09	11,000

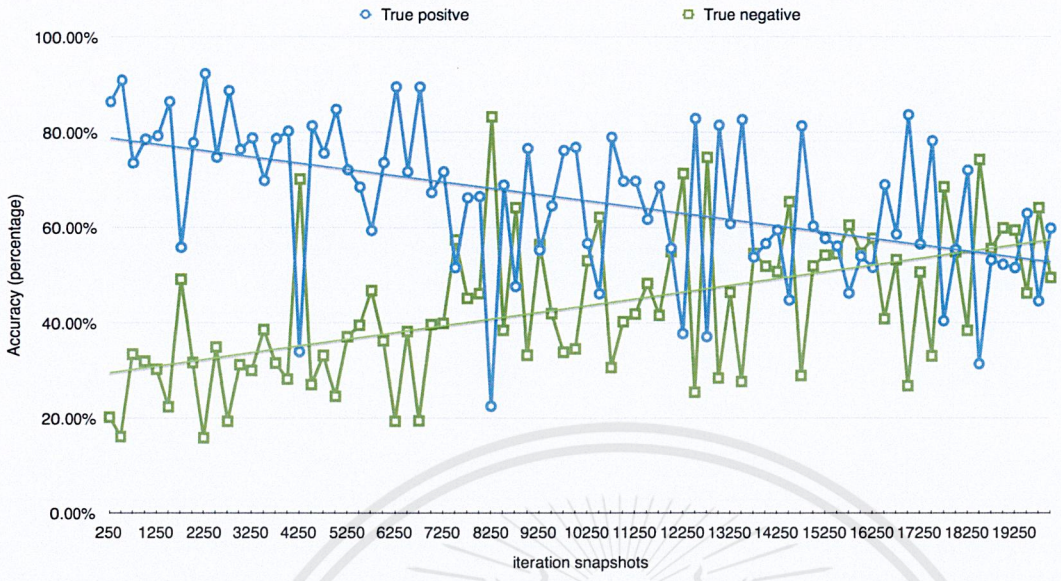


Figure 4.7: The network accuracy in each snapshot. True positive percentage is a blue dot on a graph while green dot represent false negative percentage.

4.3.3 Experiment 3: Binary CNN classification with training region selection

Objective

New binary CNNs for each target class are trained with a preprocessed data samples. Also the number of samples are increased by extracting blocks from the leaf images as overlapping blocks. This is to solve the mirror problem in the previous experiment. The experiment was conducted to measure the performance of the newly trained binary CNNs.The overall process is the same as the experiment in Section 4.3.2 and is depicted in Fig. 4.5.

Experiment setup

Blocks are re-extracted from the leaf images in the dataset. Each extracted block overlaps with its adjacent block in the leaf image. Before feeding these newly extracted blocks into each binary CNN for training, all blocks are required to be pre-

processed with histogram analysis first. This method is used to filtered out blocks from deficient leaves that appear to be healthy. The deficient blocks that are filtered out will not be used for training. More details about the histogram analysis method used is explained in Section 3.3.2. With this reason it makes binary classification can't be able to classify blocks correctly. For dataset selection, This experiment selects 80 percent, 10 percent and 10 percent for training data, testing data and validation data respectively.

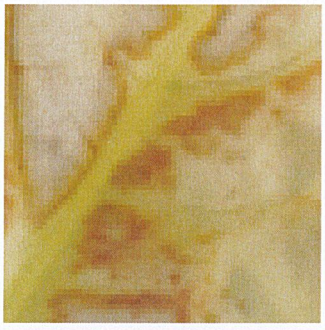
Dataset selection: Some plants are removed from the dataset for this experiment. This is due to different lightning, and the plants conditions. The numbers of the removed plants for each class are shown in Table 4.10.

Table 4.10: The plant number that has removed for each class.

Class	Removed plant no.
Healthy	2, 7
Calcium (-Ca)	12
Iron (-Fe)	10, 12
Magnesium (-Mg)	N/A
Nitrogen (-N)	5, 9
Potassium (-K)	5, 9

Preprocessing step: As mention earlier in this section, the experiment uses histogram analysis to filter out deficient blocks without abnormalities presented. Examples of the deficient blocks with and without abnormalities are shown in Fig. 4.8. Blocks are also extracted with size 64×64 pixels. The overlapping region is half of the block size which is 32 pixels. Using overlapping blocks, the number of samples is significantly increased. Table 4.11 shows the number of preprocessed blocks for each class.

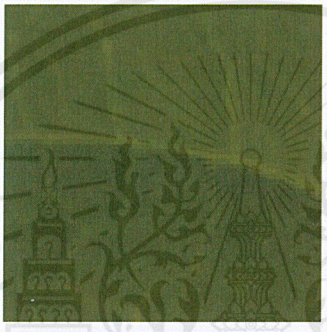
(a) Calcium deficient block with abnormal region



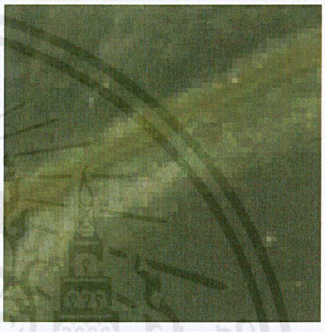
(b) Iron deficient block with abnormal region

(c) Magnesium deficient block with abnormal region

(d) Calcium deficient block without abnormal region

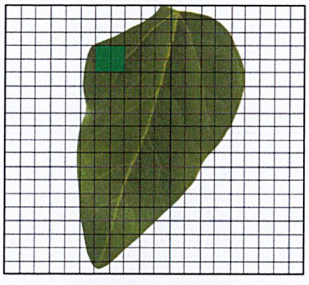


(e) Iron deficient block without abnormal region

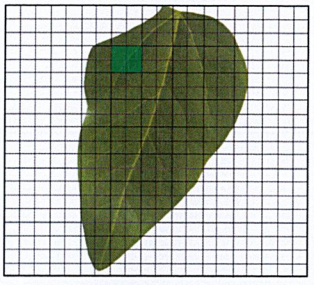


(f) Magnesium deficient block without abnormal region

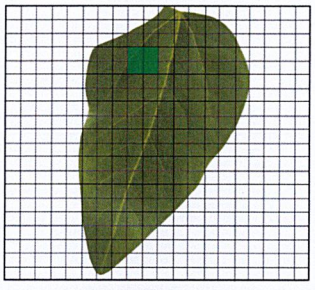
Figure 4.8: Examples of deficient blocks with (first row) and without (second row) abnormalities. The blocks are labeled as deficient but some blocks do not have symptom occurred



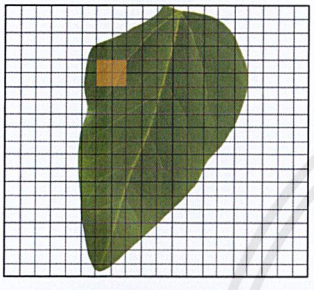
(a) Select the block to crop in first row



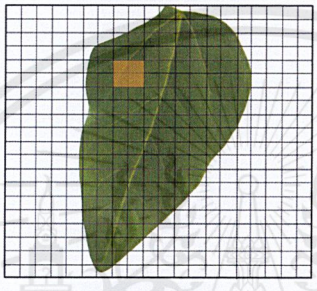
(b) step the block 32 pixels and select the second block



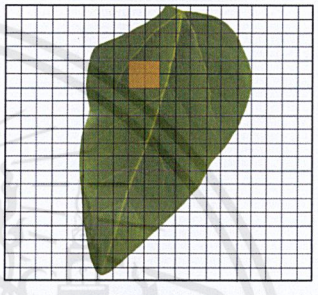
(c) step the block 32 pixels and select the third block



(d) Select the block to crop in second row by step down 32 pixels



(e) step the block 32 pixels and select the second block



(f) step the block 32 pixels and select the third block

Figure 4.9: Examples of extracting the image block by overlapping the other block with 32 pixels.

Table 4.11: The number of block samples are used in the experiment.

Nutrient deficiency	Number of blocks		
	Train	Validate	Test
Healthy	29,618	0	6,894
Calcium (Ca)	706	174	258
Iron (Fe)	14,333	3,592	162
Magnesium (Mg)	4,990	592	1,581
Nitrogen (N)	6,123	1,054	1,707
Potassium (K)	1,789	674	310
Total	**57,559**	**3,934**	**13,064**

Parameter setting: Table 4.12 describes parameter configuration for each binary CNN. The parameters are quite different from the previous experiment because the previous experiment uses AlexNet's proposed parameters for the network but for

this experiment, they are adjusted to be more suitable with the dataset.

Table 4.12: The selected parameter setting for all binary CNNs.

Parameter name	Value
Batch size (blocks)	1
Create snapshot (iterations)	1000
Learning policy (type)	fixed
Learning rate α	0.0001
Number of iteration (iterations)	400,000
Weight decay	0.004

Convolutional neural network architecture and parameters: A convolutional neural network is used in this experiment. The network is the same as in the previous experiment. The only difference is the increased number of sample blocks and the preprocessing (histogram analysis) of them. To give an example, as depicted in Fig. 4.10, with histogram analysis, only the red blocks are used to train the network.

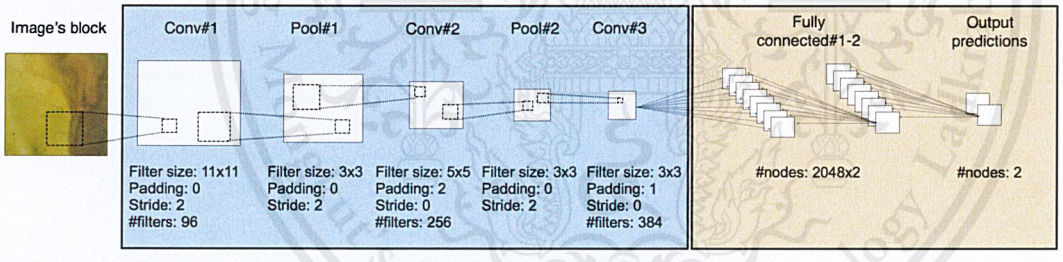


Figure 4.10: The flow of process and network parameters which are used in this experiment

Result and discussion

The classification accuracy and time used to train the network of each binary CNN are shown in Table 4.13. In this experiment, there are two models of graphic cards used which are Nvidia GeForce GTX 960 and GeForce GTX 970. Binary classification for calcium, magnesium and nitrogen is training with GeForce GTX 970. Using GTX 970 reduces the training time for about 33 % comparing to training

with GTX 960. For the result, overall, it seems all binary CNNs achieve quite good performance with above 70% accuracies. The network for healthy class obtains the highest in all the measures used. Iron classifier is the second best with 74.65% f-measure. The rest of the classifiers perform quite poorly and receive around 20% of f-measure except for calcium classifier which achieves even poorer result with less than 10 % f-measure. Additionally, these low f-measure values resulted from low recall since all classifiers receive relatively high precision. With additional preprocessing, classification performance improves significantly. However, it also leads lack of training samples for calcium and potassium deficiency classes. The reason for this is that we had quite low number of training samples for calcium already before applying histogram analysis. For the case of potassium, the symptom usually appears at leaves's tips which hardly be extracted out from a leaf image since we only extract blocks containing no background region.

Table 4.13: The classification performance and training time of each binary CNN.

Binary CNN	Precision (%)	Recall (%)	F-measure (%)	Accuracy (%)	Time (minutes)
Healthy	90.80	96.77	93.69	83.64	170
Calcium (-Ca)	64.34	4.31	8.08	70.92	110
Iron (-Fe)	81.43	68.91	74.65	80.08	175
Magnesium (-Mg)	87.84	11.93	21.01	79.51	106
Nitrogen (-N)	59.68	17.14	26.63	79.62	104
Potassium (-K)	84.13	11.49	20.22	73.98	174
Average	**78.04**	**35.09**	**48.41**	**77.96**	**140**

4.3.4 Experiment 4: Compare one-vs-all block-level classification using MLP with winner-takes-all approach

Objective

The experiment was conducted to compare the one-vs-all classifier using the winner-takes-all and using the MLP-based to determine the deficiency presented in the block-level . One-vs-all classifier is a classifier resulting from combining all the binary CNNs together. The overall process can be seen in Fig. 4.11.

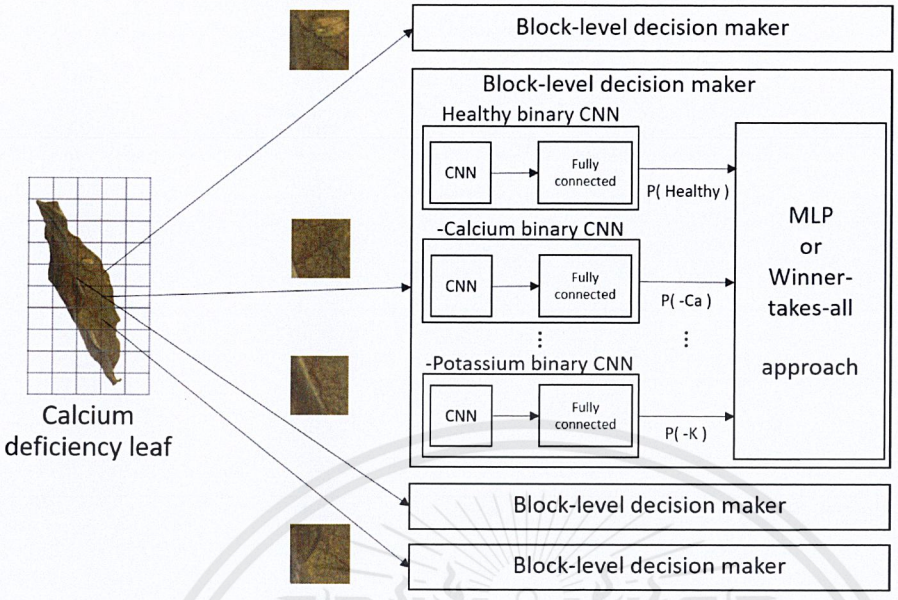


Figure 4.11: The flow of process and network parameter which are used in this experiment

Experiment setup

At first, a block is fed to each binary CNN as in Section 4.3.2. The output probabilities from each binary CNN are fed to the two one-vs-all classifiers and the results are compared. The first one-vs-all classifier uses winner-takes-all approach which simply outputs the class having highest probability as the decision about the deficiency presented in the block. The output probabilities are also fed to another one-vs-all classifier based on MLP which uses ELM as a learning procedure. The block is then assigned the nutrient deficiency class that produces highest output signal from the MLP. The training set is a set of six-probabilities resulting from running the classification in Section 4.3.2 on the training data and the testing set is a set of six-probabilities resulting from the same classification on testing data.

Neural network: The classifier used in the experiment is a MLP trained with ELM. The network has one hidden layer with seven hidden nodes. The network has six output nodes corresponding to each nutrient deficiency class.

Result and discussion

The classification accuracy of each class of deficiency for each approach is shown in Tables 4.14 and 4.15. Overall, the performance for both approach are poor with only around 20% of precision and recall. However, for MLP approach, the classes that obtain better performance are quite significantly better than those of winner-takes-all approach. The best recognized class are the same for both approaches which is iron deficiency class. . The class which receives the poorest performance for both approaches is potassium. The reason for this is beacuse symptom of potassium deficiency usually appears on a leaf tip which hardly be included in the extracted block images. Therefore, the blocks used to train the classifier do not represent the symptom of the deficiency which consequently leads to bad classification performance.

Table 4.14: The performance of block-level one-vs-all classification using MLP.

Class	Precision (%)	Recall (%)	F-measure (%)
Healthy	21.32	38.17	27.36
Calcium (-Ca)	21.24	24.30	22.67
Iron (-Fe)	40.41	36.99	38.62
Magnesium (-Mg)	19.47	24.01	21.50
Nitrogen (-N)	18.80	15.17	16.79
Potassium (-K)	25.78	2.09	3.87
Average	**24.97**	**24.01**	**24.48**

Table 4.15: The performance of block-level one-vs-all classification using winner-takes-all.

Class	Precision (%)	Recall (%)	F-measure (%)
Healthy	19.07	17.44	18.22
Calcium (-Ca)	19.41	36.41	25.32
Iron (-Fe)	51.52	24.38	33.10
Magnesium (-Mg)	19.10	31.79	23.86
Nitrogen (-N)	25.04	10.50	14.80
Potassium (-K)	19.21	19.94	19.57
Average	**26.48**	**22.59**	**24.38**

From the result in Table 4.6, it seem to produce the accuracy result similar to the winner-takes-all approach, but the result in neural network-based approach produces a better result in both experiment in Sections 4.3.1 and 4.3.4

4.3.5 Experiment 5: Compare one-vs-all block-level classification using MLP with winner-takes-all approach with data preprocessing

Objective

As new data set from Section 4.3.3 is acquired and it performs quit well for binary classification, an experiment similar to experiment four was conducted again to improve the accuracy of block-level classification. The difference this experiment and the previous one is only the dataset which proves to be better in the case of block-level classification. The goal here is to compare the performance of one-vs-all based on MLP with winner-takes-all using new preprocessed dataset. The overall process is depicted in Fig. 4.12.

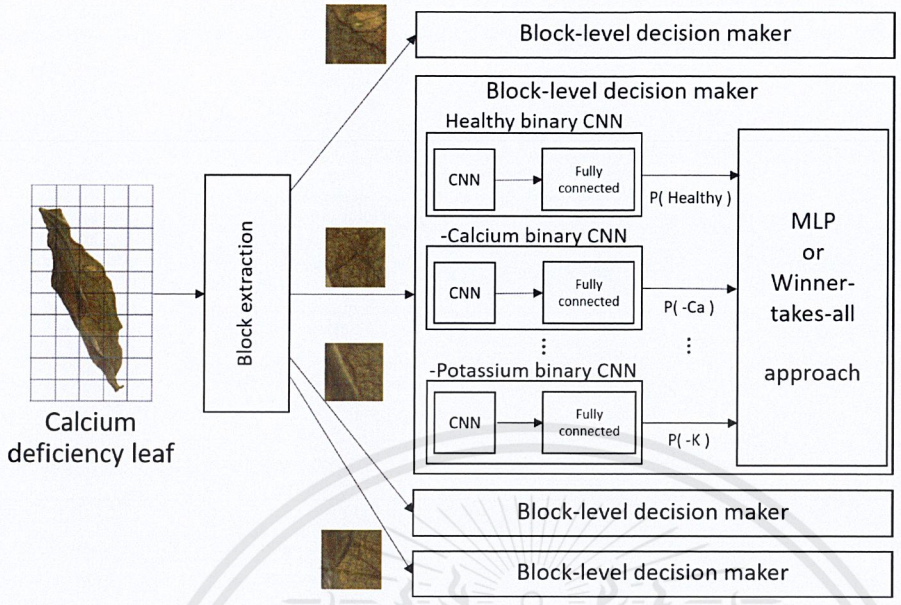


Figure 4.12: The flow of process and network parameter which are used in this experiment the input is used from the experiment in Section 4.3.3 Section 4.3.4

Experiment setup

The experiment setup is the same as in Section 4.3.4. A block is fed to each binary CNN as in the experiment in Section 4.3.3. The output probabilities from each binary CNN are then fed to MLP-based and winner-takes-all-based one-vs-all classifiers. The results are then compared. Architecture and all the parameters of the network are the same which is a MLP trained with ELM. The training set is a set of six-probabilities resulting from running the classification in Section 4.3.3 on the training data and the testing set is a set of six-probabilities resulting from the same classification on testing data.

Result and discussion

As shown in Tables 4.16 and 4.17, the overall accuracy has significantly improved comparing to the result in experiment four. To compare MLP approach with winner-takes-all approach, the two approaches yield comparable results. Healthy and iron deficient blocks are well recognized by the two approaches. However, for

MLP approach, calcium and potassium deficiency cannot be detected at all. This should be due to relatively small number of training samples for the two classes. Still, MLP based approach perform statistically significantly better for magnesium and nitrogen deficiency classes.

Table 4.16: The performance of block-level one-vs-all classification using MLP.

Class	Precision (%)	Recall (%)	F-measure (%)
Healthy	91.06	92.17	91.61
Calcium (-Ca)	0.00	0.00	0.00
Iron (-Fe)	71.61	76.89	74.16
Magnesium (-Mg)	24.62	70.44	36.49
Nitrogen (-N)	25.23	12.81	16.99
Potassium (-K)	0.00	0.00	0.00
Average	**70.89**	**74.00**	**72.41**

Table 4.17: The performance of block-level one-vs-all classification using winner-takes-all.

Class	Precision (%)	Recall (%)	F-measure (%)
Healthy	98.02	89.06	93.32
Calcium (-Ca)	6.85	13.57	9.10
Iron (-Fe)	80.58	61.33	69.84
Magnesium (-Mg)	16.80	54.90	25.73
Nitrogen (-N)	18.32	5.60	8.58
Potassium (-K)	19.86	38.28	26.15
Average	**77.28**	**69.04**	**72.93**

4.3.6 Experiment 6: Predict the nutrient deficiency of a whole leaf

Objective

The experiment was conducted to measure the accuracy of decision making regarding the deficiency of the whole leaf image using the information of each block it contains. The overall process is shown in Fig. 4.13.

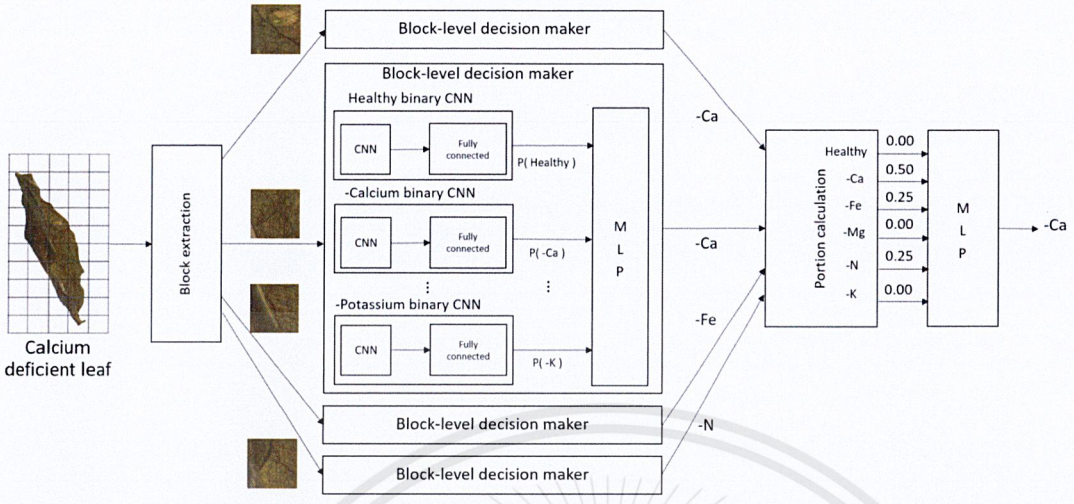


Figure 4.13: The flow of process in predicting the deficiency of whole leaf

Experiment setup

Once all the blocks in a leaf are classified using an MLP-based one-vs-all classifier as in Section 4.3.4, the portions of blocks classified as each nutrient deficiency are calculated. They are then used as inputs to another neural network. The training and testing set are acquired from running the classifications in experiment five on training and testing data respectively.

Dataset: The training set and testing set are the blocks from training and testing set in experiment five respectively. However, in this experiment, the blocks are grouped into leaves and within the same leaf, the portion of blocks of each class classified in experiment five are used as data sample in this experiment.

Neural network: MLP trained with ELM is used again in this experiment. The network contains one hidden layer with six hidden nodes. The network outputs six values for each nutrient deficiency class.

Result and discussion

The accuracy of the prediction is shown in Table 4.18. Again, calcium and potassium deficiency classes cannot be recognized by the MLP here. This is due to

the same reason as in the previous experiment that the number of training samples for the two classes are not enough. Healthy and iron deficient leaves receive the best performance which is consistent with the result from block-level classification. Magnesium deficiency also is recognized relatively well with a sensitivity (recall) value of around 70% and nitrogen deficiency receives the worst performance disregarding the two classes which cannot be recognized. Also, Table 4.18 and Fig. 4.14 show the accuracy of the system by day from day 1 to day 28. The fluctuation in the graph should be due to unequal proportion of each class in each day.

Table 4.18: The performance of leaf-level classification.

Class	Precision (%)	Recall (%)	F-measure (%)
Healthy	68.61	88.68	77.36
Calcium (-Ca)	0.00	0.00	0.00
Iron (-Fe)	64.41	61.29	62.81
Magnesium (-Mg)	24.10	66.67	35.40
Nitrogen (-N)	26.92	16.67	20.59
Potassium (-K)	0.00	0.00	0.00
Average	**43.02**	**52.13**	**47.14**

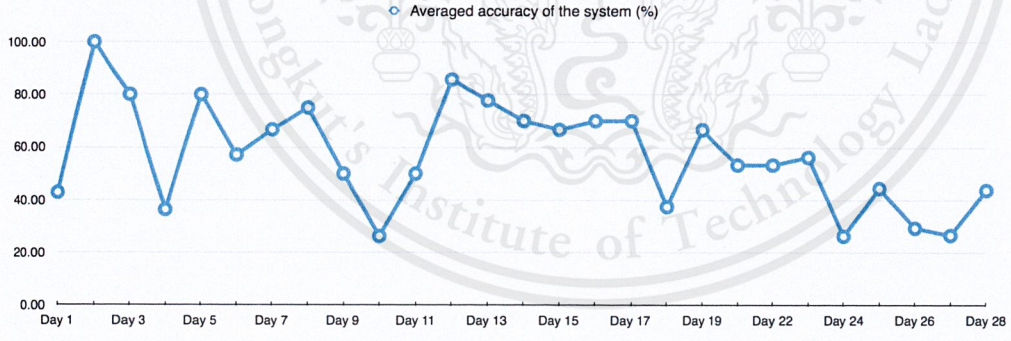


Figure 4.14: Line chart showing averaged accuracy of the system plotted by day.

Table 4.19: The performance of leaf-level classification by days.

	Averaged accuracy (%)
Day 1	42.86
Day 2	100.00
Day 3	80.00
Day 4	36.36
Day 5	80.00
Day 6	57.14
Day 7	66.67
Day 8	75.00
Day 9	50.00
Day 10	26.32
Day 11	50.00
Day 12	85.71
Day 13	77.78
Day 14	70.00
Day 15	66.67
Day 16	70.00
Day 17	70.00
Day 18	37.50
Day 19	66.67
Day 21	53.33
Day 22	53.33
Day 23	56.25
Day 24	26.32
Day 25	44.44
Day 26	29.41
Day 27	26.67
Day 28	43.75

4.3.7 Experiment 7: Measure human classification performance

Objective

The experiment was conducted to measure naked-eye human performance in classifying plant nutrient deficiencies.

Experiment setup

A web application is created to let a participant do a test. In the test, 800 leaf images are randomly shown to the participants. They then need to answer which of the six deficiencies is presented in the leaf image shown. Leaf images used in the experiment are the same as in the dataset used to test the system. Two participants participated in this experiment.

Result and discussion

The performances of each participant are shown in Table 4.20. Overall, the result from both participants yield sensible performance because it is quite tough for human to identify the symptoms in the early stage of deficiencies and to distinguish between symptoms from deficiencies which appear to be similar. The table shows that both participants usually answer that the plant is healthy due to the fact that some leaves only have quite small region of abnormalities. In consequence, healthy class receive low precision but high recall. On the other hand, iron deficiency is the easiest to detect as it leads to an quit easily distinguishable appearance comparing to other deficiencies. Hence, both participants achieved the highest precision around 77% to 94% for iron deficiency class. The class that received the worst performance from both participants is potassium deficiency. This is because the symptom only appears on a small region on the tip of a leaf.

Table 4.21 shows the averaged accuracy of both participants and Fig. 4.15 depicts the same result as a line chart. Obviously, in the early days of the deficiencies,

Table 4.20: Classification performances perform by participator one and participator two

Class	1st participator			2nd participator		
	Precision (%)	Recall (%)	F-measure (%)	Precision (%)	Recall (%)	F-measure (%)
Healthy	21.98	58.17	31.90	19.07	65.22	29.51
Calcium (-Ca)	30.59	25.00	27.51	34.21	23.01	27.51
Iron (-Fe)	77.55	27.94	41.08	93.75	12.82	22.56
Magnesium (-Mg)	29.79	20.44	24.24	38.46	23.44	29.13
Nitrogen (-N)	28.05	18.55	22.33	23.38	11.46	15.38
Potassium (-K)	21.18	12.33	15.58	17.28	9.52	12.28
Average	**34.68**	**27.75**	**27.12**	**35.75**	**24.13**	**22.21**

it is quite difficult for a human to detect the deficiencies because they haven't appeared or hardly presented. However, in the later days, The abnormalities start to more obviously appear on leaves and are easier to be detected by human. Fluctuation of the graph is resulted from unequally presented abnormalities on each leaf. Both participants achieve the best accuracy around day 18 to day 26. Additionally, the confusion matrices for each day is also provided (see appendix B).

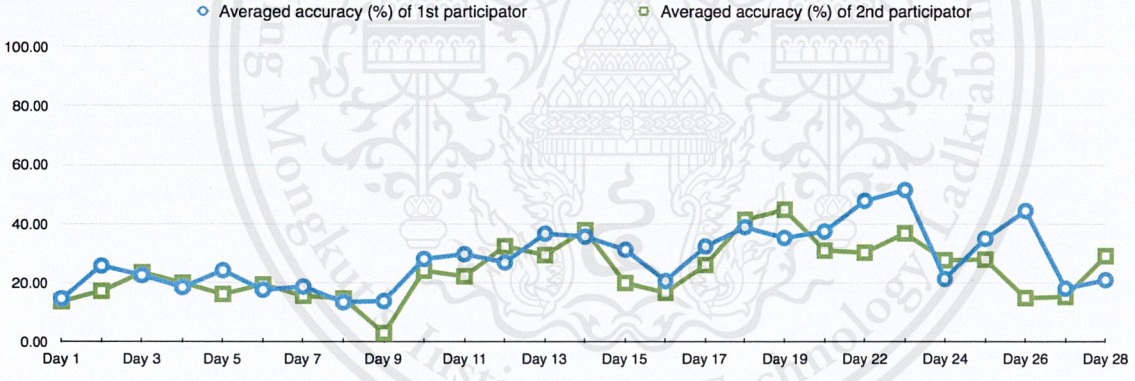


Figure 4.15: Line chart of participators averaged accuracy

Table 4.21: Human classification performance by days

	Averaged accuracy (%)	
	1^{st} participant	2^{nd} participant
Day 1	14.71	13.79
Day 2	25.81	17.24
Day 3	22.58	23.53
Day 4	18.52	20.00
Day 5	24.24	16.22
Day 6	17.65	19.44
Day 7	18.75	15.63
Day 8	13.51	14.71
Day 9	13.89	3.13
Day 10	28.13	24.14
Day 11	29.73	22.22
Day 12	26.92	32.43
Day 13	36.67	29.41
Day 14	35.71	37.84
Day 15	31.25	20.00
Day 16	20.69	16.67
Day 17	32.35	26.09
Day 18	38.89	41.38
Day 19	35.29	44.83
Day 21	37.50	31.03
Day 22	47.83	30.30
Day 23	51.52	36.84
Day 24	21.43	27.78
Day 25	35.00	28.00
Day 26	44.44	15.00
Day 27	18.18	15.38
Day 28	21.05	29.17

4.3.8 Experiment 8: Compare human classification performance with the system

Objective

The experiment was conducted to compare the classification performance of the human and the system.

Experiment setup

The system classification performance in Section 4.3.6 is used to compare with the human classification performance in Section 4.3.7.

Result and discussion

Obviously, for calcium and potassium deficiencies, although human achieve quite low performance, the performance is much better than those of the system (which cannot recognize these classes at all due to insufficient training samples). However, overall, the system still achieve higher performance as can be seen in Fig. 4.16 and disregarding calcium and potassium deficiencies, the system achieve much higher performance in all of the classes. Table 4.22 compares the the system with the averaged human performance computed by averaging the two participants performance. Therefore, on average, the proposed system significantly outperforms human ability in recognizing plant nutrient deficiencies from leaf images.

Table 4.22: Comparison between the performance of the system and the averaged performance of the two participants in Section 4.3.6.

Class	System performance			Averaged human performance		
	Precision (%)	Recall (%)	F-measure (%)	Precision (%)	Recall (%)	F-measure (%)
Healthy	68.61	88.68	77.36	15.03	61.70	24.00
Calcium (-Ca)	0.00	0.00	0.00	24.75	24.01	23.24
Iron (-Fe)	64.41	61.29	62.81	66.27	20.38	27.52
Magnesium (-Mg)	24.10	66.67	35.40	26.68	21.94	23.18
Nitrogen (-N)	26.92	16.67	20.59	18.70	15.01	15.68
Potassium (-K)	0.00	0.00	0.00	13.94	10.93	11.84
Average	**43.02**	**52.13**	**47.14**	**27.37**	**26.46**	**20.85**

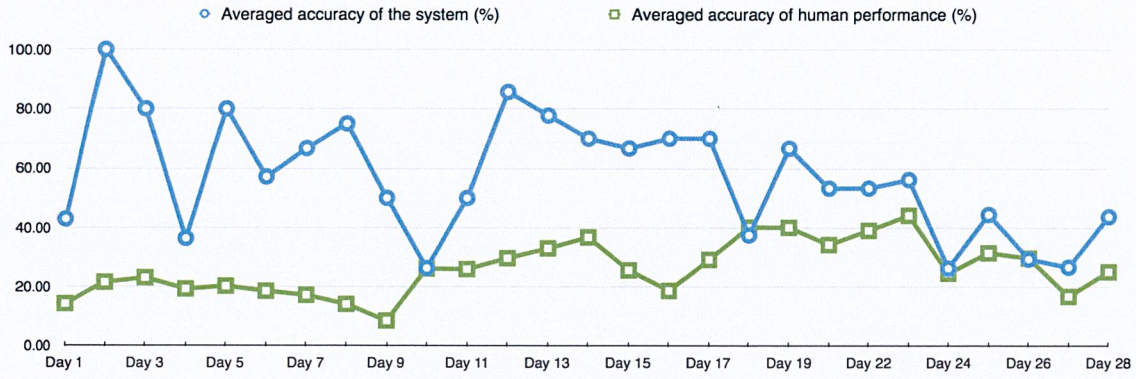


Figure 4.16: Line chart of the averaged accuracy of the system (blue line) and human (green line) plotted by day

Chapter 5

Conclusion

The thesis proposes a plant nutrient deficiency classification system. The system is used to detect deficiencies in plants. It allows the user to take an image with a mobile application and the system will classify the deficiency presented in the input leaf image. The system then returns the result back to the user. The system is separated into two parts, which are mobile application (Section 3.1) and server (Section 3.2).

In the mobile application, the image segmentation called active contour model (Section 2.2) is used to segment the acquired leaf image from background region after the acquisition process. Then the application sends the segmented image to the server. After server receives the segmented image, the server calls the recognition component (Section 3.3) to classify the deficiency in the leaf image.

In the recognition subsystem, the leaf in an image will be cropped into multiple blocks and fed to six binary convolutional neural network or CNN (Section 2.4). The experiment on this procedure can be found in Section 4.3.3.Then the result from each classifier will be used to determine the deficiency in each block using a multi-layer perceptron or MLP (Section 2.3) learned using extreme learning machine or ELM (Section 2.7). The experiment on this process is explained in Section 4.3.5. The last step of recognition subsystem is to used the classification results from each

block to make a decision regarding the deficiency of the leaf image by using another MLP trained using ELM.

The experiment in Section 4.3.6 summarizes the performance of the system. The experiment in Section 4.3.7 evaluates human performance in doing the same task. The experiment in Section 4.3.8 compares the performance of the system with human and concludes that on average, the proposed system outperforms human in recognizing nutrient deficiencies in plants using leaf images.

5.1 Future work

5.1.1 Mobile application

There is still a room to improve the mobile application. The segmentation algorithm used in the system can still be improved. The system also can still be improved. For example, the recognition system might be implemented to be fully embedded within the mobile application, hence, the connection between the mobile application and the server will not be required.

5.1.2 Recognition component

There are still plenty of techniques which can be used to improve the recognition component such as ensemble method [3] which, basically, is a way to improve recognition performance by combining multiple classifiers together. Many other architecture of CNN can be deployed and parameters of the network learning procedures can also be adjusted in numerous ways. Additionally, more data set can be used to improve classification performance which is the easiest way to improve classification performance.

Bibliography

[1] "Panomex Inc," available at http://www.panomex.com/plant-test-meters/plant-nutrition-analyzer.html, last accessed: Oct 18, 2016.

[2] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[3] Dietterich, Thomas G, "Ensemble methods in machine learning," International workshop on multiple classifier systems, Springer, pp. 1-15, 2000.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[5] S. C. Hodges and G. Constable, "Plant responses to mineral deficiencies and toxicities," Physiology of Cotton, pp. 142–161, 2010.

[6] H. Huang, Q. Zhu and C. Siew, "Extreme learning machine: theory and applications," Neurocomputing, vol. 70,pp. 489-501, 2016.

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev and J. Long et Al., "Caffe: Convolutional architecture for fast feature embedding", ArXiv preprint arXiv:1408.5093, 2014.

[8] M. Kass, A. Witkin and D. Terzopoulos, " Snakes: Active contour models," International Journal of Computer Vision, vol. 1, pp. 321-331, 1988.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Proceedings of 25th Advances in Neural Information Processing Systems (NIPS 2012), pp. 1097–1105, 2012.

[10] B. Rangel, M. Fernández, J.C. Murillo, J.C.P. Ortega and J.M.R. Arreguín, "KNN-based image segmentation for grapevine potassium deficiency diagnosis," Proceeding of the 2016 International Conference on Electronics, Communications and Computers (CONIELECOMP), pp.48-53, 2016.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, pp. 436–444, 2015.

[12] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol.86, no.11 pp. 2278-2324, 1998.

[13] A. McAndrew, An Introduction To Digital Image Processing With MATLAB, Course Technology Press, Boston, MA, United States, 2004.

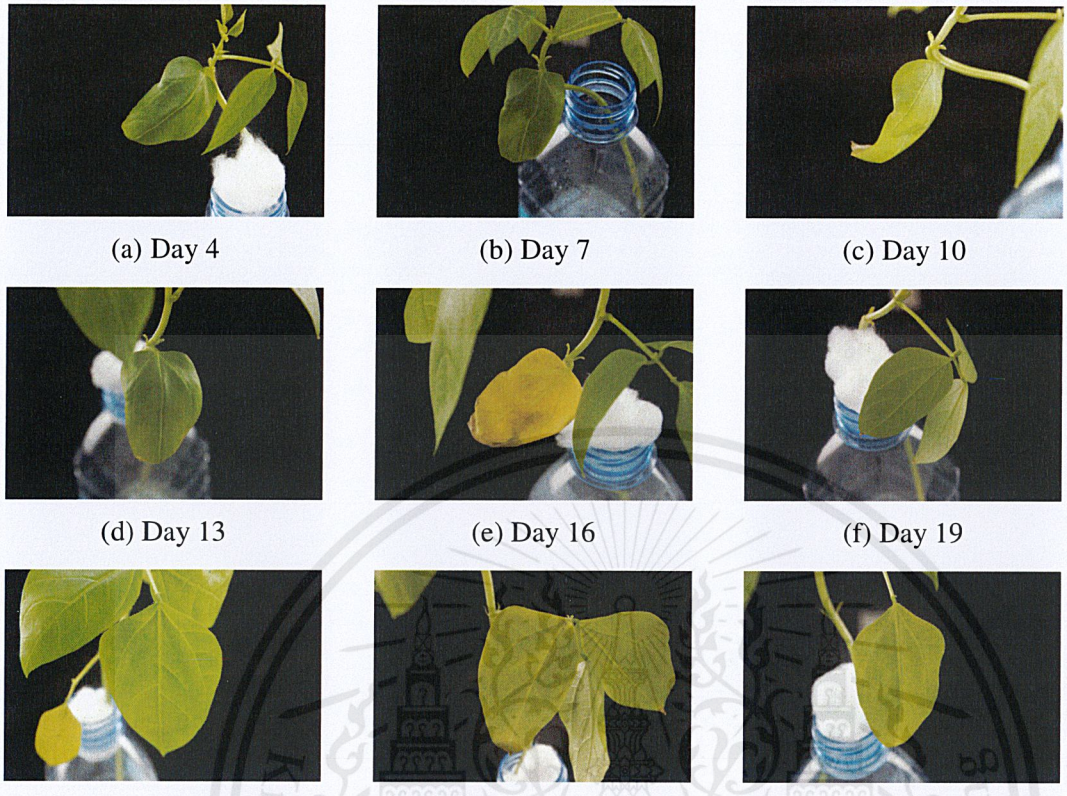
[14] U. Mokhtar, M. Ali, and H. Hefny, "Tomato leaves diseases detection approach based on support vector machines," Proceedings of 2015 11th International Computer Engineering Conference (ICENCO), pp.246-250, 2015.

Appendices

Appendix A: Leaf image dataset

The examples of leaf image dataset are shown in Fig.A.1 - A.12. As mention in experiment section, the dataset consist of six classes and leaf at center of an image. Chapter A describes the number of leaf images belong to each captured day from day 1 to day 28.

Figure A.1: An example of a healthy plant leaf (day 1).



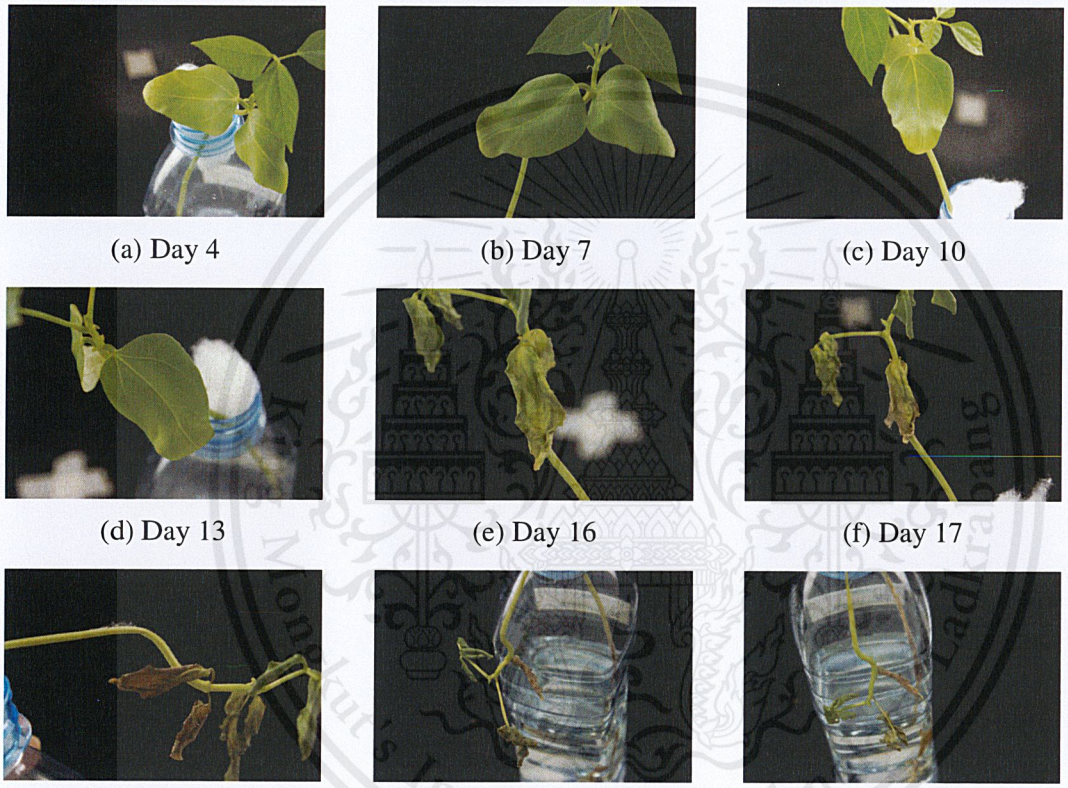
(a) Day 4 (b) Day 7 (c) Day 10

(d) Day 13 (e) Day 16 (f) Day 19

(g) Day 23 (h) Day 26 (i) Day 28

Figure A.2: Examples of healthy plant leaf from day 1 to day 28.

Figure A.3: An example of a calcium deficient plant leaf (day 1).



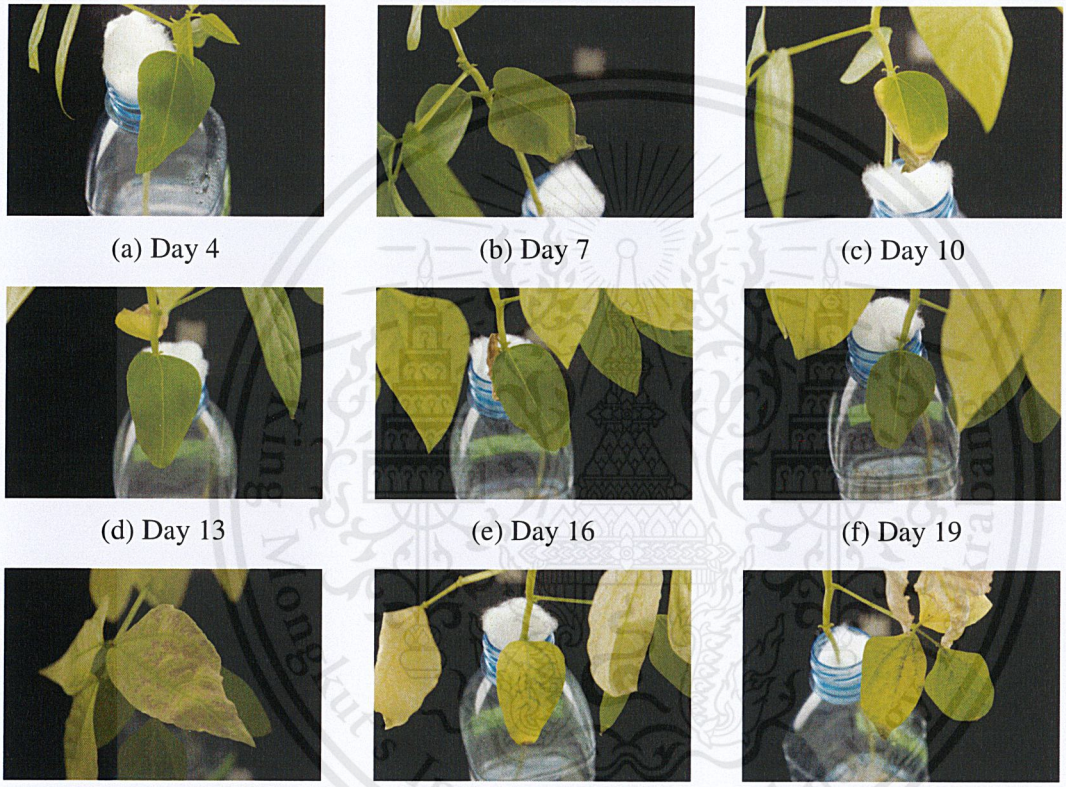
(a) Day 4 (b) Day 7 (c) Day 10

(d) Day 13 (e) Day 16 (f) Day 17

(g) Day 18 (h) Day 19 (i) Day 21

Figure A.4: Examples of calcium deficient plant leaves from day 1 to day 28.

Figure A.5: An example of an iron deficient plant leaf (day 1).



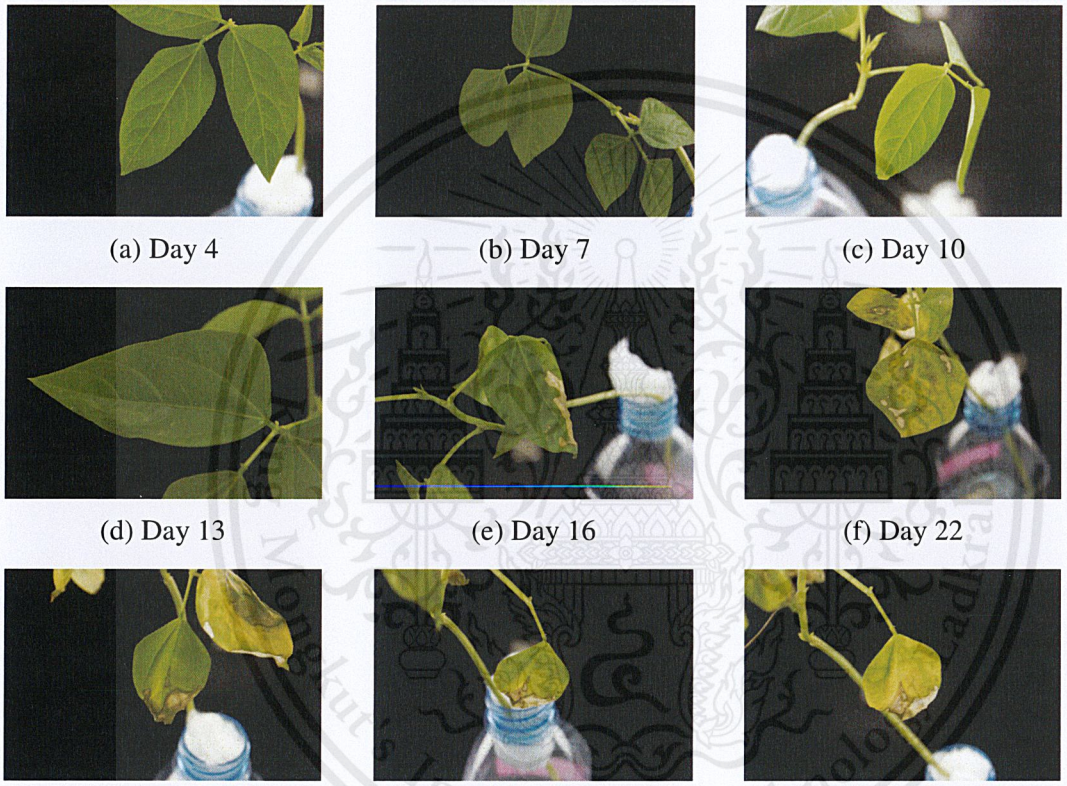
(a) Day 4 (b) Day 7 (c) Day 10

(d) Day 13 (e) Day 16 (f) Day 19

(g) Day 23 (h) Day 26 (i) Day 28

Figure A.6: Examples of iron deficient plant leaves from day 1 to day 28.

Figure A.7: An example of a magnesium deficient plant leaf (day 1).



(a) Day 4 (b) Day 7 (c) Day 10

(d) Day 13 (e) Day 16 (f) Day 22

(g) Day 23 (h) Day 26 (i) Day 28

Figure A.8: Examples of magnesium deficient plant leaves from day 1 to day 28.

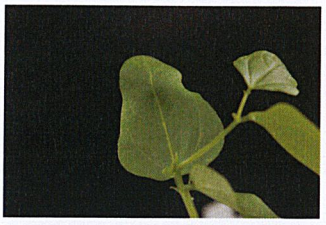
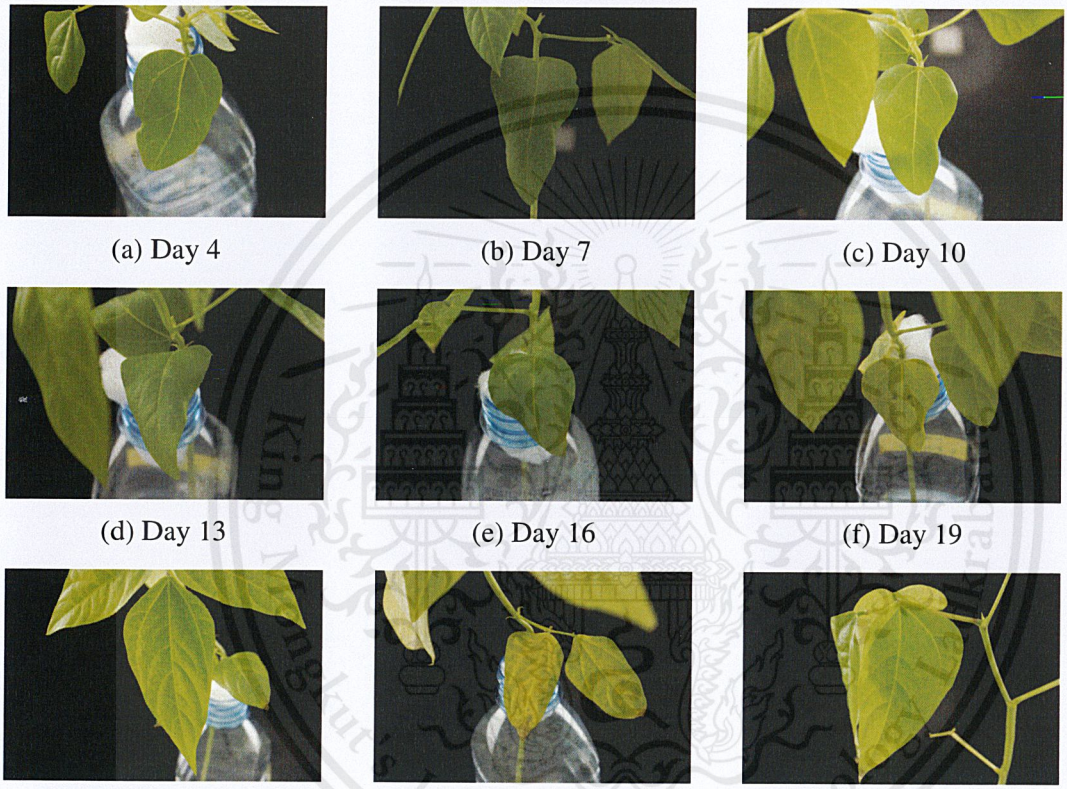


Figure A.9: An example of a nitrogen deficient plant leaf (day 1).



(a) Day 4 (b) Day 7 (c) Day 10

(d) Day 13 (e) Day 16 (f) Day 19

(g) Day 23 (h) Day 26 (i) Day 28

Figure A.10: Examples of nitrogen deficient plant leaves from day 1 to day 28.

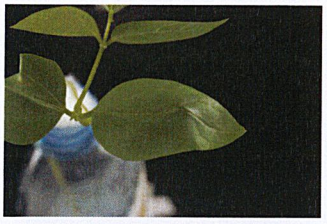
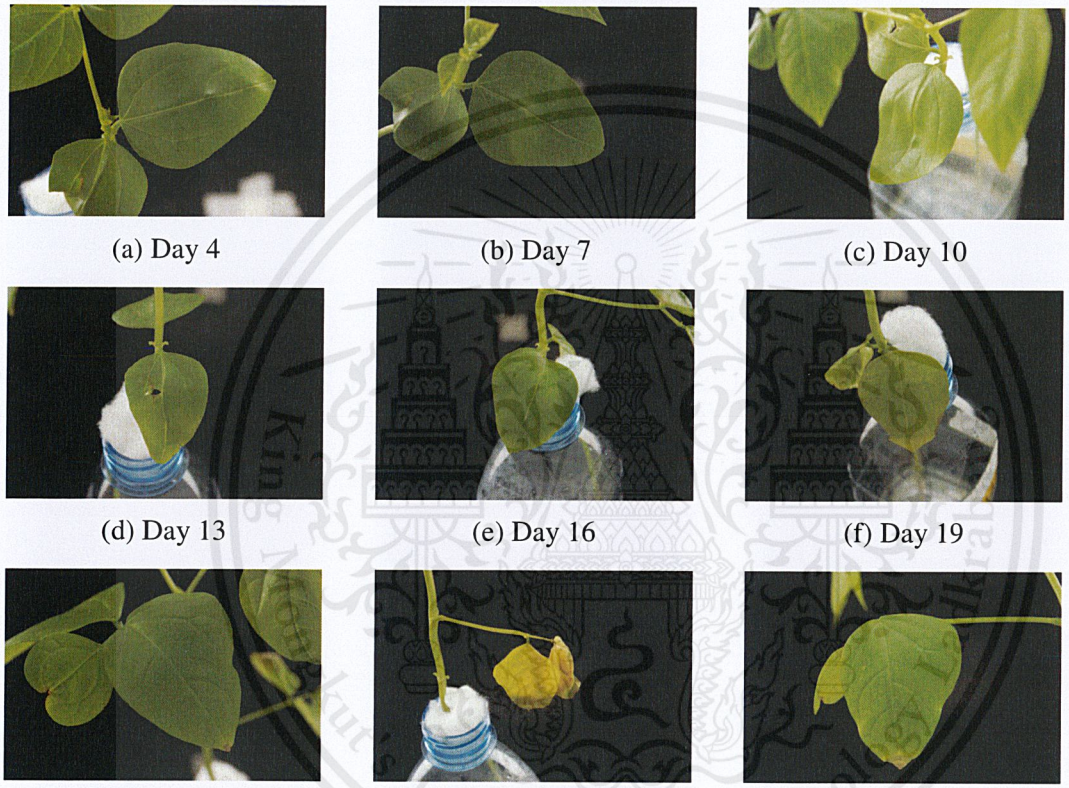


Figure A.11: An example of a potassium deficient plant leaf (day 1).



(a) Day 4 (b) Day 7 (c) Day 10

(d) Day 13 (e) Day 16 (f) Day 19

(g) Day 23 (h) Day 26 (i) Day 28

Figure A.12: Examples of potassium deficient plant leaves from day 1 to day 28.

Table A.1: The number of deficient leaf images taken in 28 days

Class name	Day																											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Healthy	22	23	24	24	24	24	24	23	24	24	24	24	24	24	24	24	24	24	24	0	23	24	23	23	21	21	19	19
Calcium (-Ca)	23	23	23	23	24	24	24	24	24	22	22	20	18	18	18	18	18	18	18	0	18	6	6	6	7	7	7	7
Iron (-Fe)	20	20	21	21	21	21	23	24	24	23	23	23	23	23	23	22	22	22	21	0	21	22	23	23	23	21	19	19
Magnesium (-Mg)	20	20	22	22	23	23	23	23	23	22	22	22	22	24	24	23	22	22	22	0	22	23	23	23	21	21	18	18
Nitrogen (-N)	22	22	22	22	24	24	24	24	24	24	24	24	24	24	23	23	23	23	23	0	17	21	21	21	19	18	17	17
Phosphorus (-P)	19	19	19	18	21	21	23	23	23	23	23	23	23	24	23	23	23	22	22	0	19	24	24	23	24	23	21	21
Potassium (-K)	20	20	22	22	23	23	24	24	24	24	24	24	24	24	24	24	24	24	24	0	24	23	23	23	24	22	22	22

Appendix B: Participants' classification performance

The confusion matrices perform by both participants are separate in the 28 days in Sections B.1 to B.28. In each section, there are 2 confusion matrix which perform by the first participant and the second participant respectively.

B.1 Day 1

Table B.1: Confusion matrix of the first participant on day 1

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	0	0	0	0	0	**5**	**100.00**
	-Ca	3	**0**	0	0	0	0	**3**	**0.00**
	-Fe	6	0	**0**	0	0	0	**6**	**0.00**
	-Mg	7	0	0	**0**	0	1	**8**	**0.00**
	-N	5	0	0	1	**0**	0	**6**	**0.00**
	-K	6	0	0	0	0	**0**	**6**	**0.00**
	Total	**32**	**0**	**0**	**1**	**0**	**1**	**34**	
	Precision (%)	**15.62**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

Table B.2: Confusion matrix of the second participant on day 1

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	11	**0**	0	0	0	0	**11**	**0.00**
	-Fe	5	0	**0**	0	0	0	**5**	**0.00**
	-Mg	1	0	0	**0**	0	0	**1**	**0.00**
	-N	5	0	0	0	**0**	0	**5**	**0.00**
	-K	3	0	0	0	0	**0**	**3**	**0.00**
	Total	**29**	**0**	**0**	**0**	**0**	**0**	**29**	
	Precision (%)	**13.79**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

B.2 Day 2

Table B.3: Confusion matrix of the first participant on day 2

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**8**	0	0	0	0	0	**8**	**100.00**
	-Ca	4	**0**	0	1	0	1	**6**	**0.00**
	-Fe	4	0	**0**	0	0	0	**4**	**0.00**
	-Mg	4	0	0	**0**	0	0	**4**	**0.00**
	-N	1	1	0	1	**0**	0	**3**	**0.00**
	-K	6	0	0	0	0	**0**	**6**	**0.00**
	Total	**27**	**1**	**0**	**2**	**0**	**1**	**31**	
	Precision (%)	**29.63**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

Table B.4: Confusion matrix of the second participant on day 2

		Predicted						
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	0	0	0	0	0	**5**	**100.00**
	-Ca	5	**0**	0	0	1	1	**7**	**0.00**
	-Fe	4	0	**0**	0	1	0	**5**	**0.00**
	-Mg	2	0	0	**0**	0	0	**2**	**0.00**
	-N	5	0	0	0	**0**	0	**5**	**0.00**
	-K	5	0	0	0	0	**0**	**5**	**0.00**
	Total	**26**	**0**	**0**	**0**	**2**	**1**	**29**	
	Precision (%)	**19.23**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

B.3 Day 3

Table B.5: Confusion matrix of the first participant on day 3

		Predicted						
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	1	0	0	0	0	**5**	**80.00**
	-Ca	4	**0**	0	1	0	0	**5**	**0.00**
	-Fe	3	1	**0**	1	0	0	**5**	**0.00**
	-Mg	5	0	0	**2**	0	0	**7**	**28.57**
	-N	5	0	0	0	**1**	0	**6**	**16.67**
	-K	2	0	0	1	0	**0**	**3**	**0.00**
	Total	**23**	**2**	**0**	**5**	**1**	**0**	**31**	
	Precision (%)	**17.39**	**0.00**	**0.00**	**40.00**	**100.00**	**0.00**		

Table B.6: Confusion matrix of the second participant on day 3

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**8**	0	0	0	0	0	**8**	**100.00**
	-Ca	5	**0**	0	0	0	0	**5**	**0.00**
	-Fe	3	0	**0**	0	0	0	**3**	**0.00**
	-Mg	4	0	0	**0**	0	0	**4**	**0.00**
	-N	8	0	0	0	**0**	0	**8**	**0.00**
	-K	5	0	0	0	1	**0**	**6**	**0.00**
	Total	**33**	**0**	**0**	**0**	**1**	**0**	**34**	
	Precision (%)	**24.24**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

B.4 Day 4

Table B.7: Confusion matrix of the first participant on day 4

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	1	**5**	**80.00**
	-Ca	0	**0**	1	1	1	2	**5**	**0.00**
	-Fe	1	0	**0**	0	0	0	**1**	**0.00**
	-Mg	5	0	0	**0**	0	0	**5**	**0.00**
	-N	3	0	0	0	**1**	0	**4**	**25.00**
	-K	6	0	0	1	0	**0**	**7**	**0.00**
	Total	**19**	**0**	**1**	**2**	**2**	**3**	**27**	
	Precision (%)	**21.05**	**0.00**	**0.00**	**0.00**	**50.00**	**0.00**		

Table B.8: Confusion matrix of the second participant on day 4

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	6	**0**	0	1	0	0	**7**	**0.00**
	-Fe	3	0	**0**	1	0	0	**4**	**0.00**
	-Mg	3	0	0	**1**	0	0	**4**	**25.00**
	-N	2	0	0	0	**0**	0	**2**	**0.00**
	-K	4	0	0	0	0	**0**	**4**	**0.00**
	Total	**22**	**0**	**0**	**3**	**0**	**0**	**25**	
	Precision (%)	**18.18**	**0.00**	**0.00**	**33.33**	**0.00**	**0.00**		

B.5 Day 5

Table B.9: Confusion matrix of the first participant on day 5

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**7**	0	0	0	0	0	**7**	**100.00**
	-Ca	4	**0**	0	2	0	0	**6**	**0.00**
	-Fe	2	0	**0**	2	0	0	**4**	**0.00**
	-Mg	6	0	0	**0**	1	0	**7**	**0.00**
	-N	5	0	0	0	**1**	0	**6**	**16.67**
	-K	2	0	0	1	0	**0**	**3**	**0.00**
	Total	**26**	**0**	**0**	**5**	**2**	**0**	**33**	
	Precision (%)	**26.92**	**0.00**	**0.00**	**0.00**	**50.00**	**0.00**		

Table B.10: Confusion matrix of the second participant on day 5

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**6**	0	0	0	0	0	**6**	**100.00**
	-Ca	5	**0**	0	1	0	1	**7**	**0.00**
	-Fe	2	0	**0**	2	0	0	**4**	**0.00**
	-Mg	6	0	0	**0**	0	0	**6**	**0.00**
	-N	7	0	0	0	**0**	0	**7**	**0.00**
	-K	7	0	0	0	0	**0**	**7**	**0.00**
	Total	**33**	**0**	**0**	**3**	**0**	**1**	**37**	
	Precision (%)	**18.18**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

B.6 Day 6

Table B.11: Confusion matrix of the first participant on day 6

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	0	0	0	0	1	**6**	**83.33**
	-Ca	0	**0**	0	0	1	3	**4**	**0.00**
	-Fe	3	0	**0**	0	0	0	**3**	**0.00**
	-Mg	4	0	0	**1**	0	1	**6**	**16.67**
	-N	6	0	0	2	**0**	0	**8**	**0.00**
	-K	6	0	0	1	0	**0**	**7**	**0.00**
	Total	**24**	**0**	**0**	**4**	**1**	**5**	**34**	
	Precision (%)	**20.83**	**0.00**	**0.00**	**25.00**	**0.00**	**0.00**		

Table B.12: Confusion matrix of the second participant on day 6

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	0	0	1	2	0	**8**	**62.50**
	-Ca	7	**0**	0	1	0	0	**8**	**0.00**
	-Fe	3	0	**0**	1	0	0	**4**	**0.00**
	-Mg	5	0	0	**2**	0	0	**7**	**28.57**
	-N	5	0	0	1	**0**	0	**6**	**0.00**
	-K	3	0	0	0	0	**0**	**3**	**0.00**
	Total	**28**	**0**	**0**	**6**	**2**	**0**	**36**	
	Precision (%)	**17.86**	**0.00**	**0.00**	**33.33**	**0.00**	**0.00**		

B.7 Day 7

Table B.13: Confusion matrix of the first participant on day 7

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	0	0	1	0	0	**6**	**83.33**
	-Ca	5	**0**	0	2	0	1	**8**	**0.00**
	-Fe	4	0	**1**	0	1	0	**6**	**16.67**
	-Mg	1	0	0	**0**	0	2	**3**	**0.00**
	-N	3	0	0	1	**0**	0	**4**	**0.00**
	-K	4	0	0	1	0	**0**	**5**	**0.00**
	Total	**22**	**0**	**1**	**5**	**1**	**3**	**32**	
	Precision (%)	**22.73**	**0.00**	**100.00**	**0.00**	**0.00**	**0.00**		

Table B.14: Confusion matrix of the second participant on day 7

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	3	**0**	0	1	0	0	**4**	**0.00**
	-Fe	5	0	**1**	1	0	0	**7**	**14.29**
	-Mg	5	0	0	**0**	0	0	**5**	**0.00**
	-N	8	0	0	0	**0**	0	**8**	**0.00**
	-K	3	0	0	0	1	**0**	**4**	**0.00**
	Total	**28**	**0**	**1**	**2**	**1**	**0**	**32**	
	Precision (%)	**14.29**	**0.00**	**100.00**	**0.00**	**0.00**	**0.00**		

B.8 Day 8

Table B.15: Confusion matrix of the first participant on day 8

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	1	0	2	0	1	**7**	**42.86**
	-Ca	2	**1**	0	0	0	2	**5**	**20.00**
	-Fe	4	2	**0**	0	0	0	**6**	**0.00**
	-Mg	3	0	0	**0**	1	0	**4**	**0.00**
	-N	7	0	0	2	**1**	0	**10**	**10.00**
	-K	3	0	0	2	0	**0**	**5**	**0.00**
	Total	**22**	**4**	**0**	**6**	**2**	**3**	**37**	
	Precision (%)	**13.64**	**25.00**	**0.00**	**0.00**	**50.00**	**0.00**		

Table B.16: Confusion matrix of the second participant on day 8

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	0	0	1	0	0	**6**	**83.33**
	-Ca	2	**0**	0	3	0	0	**5**	**0.00**
	-Fe	8	1	**0**	0	0	1	**10**	**0.00**
	-Mg	3	0	0	**0**	0	0	**3**	**0.00**
	-N	3	0	0	1	**0**	0	**4**	**0.00**
	-K	6	0	0	0	0	**0**	**6**	**0.00**
	Total	**27**	**1**	**0**	**5**	**0**	**1**	**34**	
	Precision (%)	**18.52**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

B.9 Day 9

Table B.17: Confusion matrix of the first participant on day 9

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	1	0	1	**5**	**60.00**
	-Ca	4	**1**	0	0	0	0	**5**	**20.00**
	-Fe	3	1	**0**	1	0	0	**5**	**0.00**
	-Mg	5	0	0	**0**	0	1	**6**	**0.00**
	-N	6	0	0	1	**1**	0	**8**	**12.50**
	-K	7	0	0	0	0	**0**	**7**	**0.00**
	Total	**28**	**2**	**0**	**3**	**1**	**2**	**36**	
	Precision (%)	**10.71**	**50.00**	**0.00**	**0.00**	**100.00**	**0.00**		

Table B.18: Confusion matrix of the second participant on day 9

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	0	0	1	0	0	**2**	**50.00**
	-Ca	4	**0**	0	1	0	2	**7**	**0.00**
	-Fe	3	0	**0**	0	1	0	**4**	**0.00**
	-Mg	4	0	0	**0**	0	2	**6**	**0.00**
	-N	6	0	0	1	**0**	2	**9**	**0.00**
	-K	4	0	0	0	0	**0**	**4**	**0.00**
	Total	**22**	**0**	**0**	**3**	**1**	**6**	**32**	
	Precision (%)	**4.55**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

B.10 Day 10

Table B.19: Confusion matrix of the first participant on day 10

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	1	0	0	2	0	**7**	**57.14**
	-Ca	1	**0**	0	2	1	0	**4**	**0.00**
	-Fe	1	0	**3**	0	1	1	**6**	**50.00**
	-Mg	4	0	0	**0**	1	0	**5**	**0.00**
	-N	0	0	1	0	**2**	1	**4**	**50.00**
	-K	3	0	0	1	2	**0**	**6**	**0.00**
	Total	**13**	**1**	**4**	**3**	**9**	**2**	**32**	
	Precision (%)	**30.77**	**0.00**	**75.00**	**0.00**	**22.22**	**0.00**		

Table B.20: Confusion matrix of the second participant on day 10

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	0	2	1	**6**	**50.00**
	-Ca	0	**0**	0	0	1	3	**4**	**0.00**
	-Fe	0	0	**0**	0	3	0	**3**	**0.00**
	-Mg	4	0	0	**0**	1	1	**6**	**0.00**
	-N	3	0	0	0	**3**	0	**6**	**50.00**
	-K	2	0	0	0	1	**1**	**4**	**25.00**
	Total	**12**	**0**	**0**	**0**	**11**	**6**	**29**	
	Precision (%)	**25.00**	**0.00**	**0.00**	**0.00**	**27.27**	**16.67**		

B.11 Day 11

Table B.21: Confusion matrix of the first participant on day 11

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	1	2	0	**6**	**50.00**
	-Ca	3	**3**	0	0	0	1	**7**	**42.86**
	-Fe	4	1	**1**	2	1	0	**9**	**11.11**
	-Mg	1	1	0	**2**	0	0	**4**	**50.00**
	-N	3	0	0	0	**1**	1	**5**	**20.00**
	-K	4	1	0	0	0	**1**	**6**	**16.67**
	Total	**18**	**6**	**1**	**5**	**4**	**3**	**37**	
	Precision (%)	**16.67**	**50.00**	**100.00**	**40.00**	**25.00**	**33.33**		

Table B.22: Confusion matrix of the second participant on day 11

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**6**	0	0	2	0	0	**8**	**75.00**
	-Ca	0	**0**	0	1	0	0	**1**	**0.00**
	-Fe	4	0	**0**	0	0	0	**4**	**0.00**
	-Mg	4	2	0	**0**	0	1	**7**	**0.00**
	-N	2	0	0	1	**0**	0	**3**	**0.00**
	-K	1	0	0	1	2	**0**	**4**	**0.00**
	Total	**17**	**2**	**0**	**5**	**2**	**1**	**27**	
	Precision (%)	**35.29**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

B.12 Day 12

Table B.23: Confusion matrix of the first participant on day 12

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	2	1	0	**6**	**50.00**
	-Ca	2	**2**	0	0	0	0	**4**	**50.00**
	-Fe	3	0	**0**	0	0	1	**4**	**0.00**
	-Mg	3	0	0	**0**	0	0	**3**	**0.00**
	-N	2	0	0	1	**2**	0	**5**	**40.00**
	-K	4	0	0	0	0	**0**	**4**	**0.00**
	Total	**17**	**2**	**0**	**3**	**3**	**1**	**26**	
	Precision (%)	**17.65**	**100.00**	**0.00**	**0.00**	**66.67**	**0.00**		

Table B.24: Confusion matrix of the second participant on day 12

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	7	0	0	0	0	0	**7**	**100.00**
	-Ca	2	**1**	0	0	0	1	**4**	**25.00**
	-Fe	2	0	**1**	2	1	2	**8**	**12.50**
	-Mg	2	0	0	**0**	0	1	**3**	**0.00**
	-N	2	1	0	1	**1**	0	**5**	**20.00**
	-K	8	0	0	0	0	**2**	**10**	**20.00**
	Total	**23**	**2**	**1**	**3**	**2**	**6**	**37**	
	Precision (%)	**30.43**	**50.00**	**100.00**	**0.00**	**50.00**	**33.33**		

B.13 Day 13

Table B.25: Confusion matrix of the first participant on day 13

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	0	0	1	0	2	**8**	**62.50**
	-Ca	1	**0**	0	2	0	0	**3**	**0.00**
	-Fe	1	0	**3**	1	1	0	**6**	**50.00**
	-Mg	2	0	1	**1**	0	1	**5**	**20.00**
	-N	1	0	0	1	**1**	0	**3**	**33.33**
	-K	4	0	0	0	0	**1**	**5**	**20.00**
	Total	**14**	**0**	**4**	**6**	**2**	**4**	**30**	
	Precision (%)	**35.71**	**0.00**	**75.00**	**16.67**	**50.00**	**25.00**		

Table B.26: Confusion matrix of the second participant on day 13

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**6**	1	0	2	0	0	**9**	**66.67**
	-Ca	1	**2**	0	0	0	1	**4**	**50.00**
	-Fe	2	0	**0**	0	3	0	**5**	**0.00**
	-Mg	5	0	0	**1**	0	0	**6**	**16.67**
	-N	3	0	0	0	**1**	1	**5**	**20.00**
	-K	4	0	0	0	1	**0**	**5**	**0.00**
	Total	**21**	**3**	**0**	**3**	**5**	**2**	**34**	
	Precision (%)	**28.57**	**66.67**	**0.00**	**33.33**	**20.00**	**0.00**		

B.14 Day 14

Table B.27: Confusion matrix of the first participant on day 14

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	1	**5**	**80.00**
	-Ca	2	**3**	0	0	0	0	**5**	**60.00**
	-Fe	2	0	**1**	0	0	0	**3**	**33.33**
	-Mg	2	0	0	**2**	0	1	**5**	**40.00**
	-N	5	0	0	0	**0**	0	**5**	**0.00**
	-K	4	0	0	1	0	**0**	**5**	**0.00**
	Total	**19**	**3**	**1**	**3**	**0**	**2**	**28**	
	Precision (%)	**21.05**	**100.00**	**100.00**	**66.67**	**0.00**	**0.00**		

Table B.28: Confusion matrix of the second participant on day 14

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	1	0	2	**7**	**57.14**
	-Ca	1	**4**	0	0	0	0	**5**	**80.00**
	-Fe	1	0	**1**	0	2	1	**5**	**20.00**
	-Mg	5	0	0	**4**	0	1	**10**	**40.00**
	-N	2	1	0	0	**0**	1	**4**	**0.00**
	-K	3	0	0	0	2	**1**	**6**	**16.67**
	Total	**16**	**5**	**1**	**5**	**4**	**6**	**37**	
	Precision (%)	**25.00**	**80.00**	**100.00**	**80.00**	**0.00**	**16.67**		

B.15 Day 15

Table B.29: Confusion matrix of the first participant on day 15

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	1	1	0	1	1	**7**	**42.86**
	-Ca	0	**2**	0	0	0	1	**3**	**66.67**
	-Fe	0	0	**1**	2	0	0	**3**	**33.33**
	-Mg	2	2	0	**1**	1	2	**8**	**12.50**
	-N	2	0	0	0	**2**	0	**4**	**50.00**
	-K	5	0	0	1	0	**1**	**7**	**14.29**
	Total	**12**	**5**	**2**	**4**	**4**	**5**	**32**	
	Precision (%)	**25.00**	**40.00**	**50.00**	**25.00**	**50.00**	**20.00**		

Table B.30: Confusion matrix of the second participant on day 15

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**2**	0	0	0	0	1	**3**	**66.67**
	-Ca	1	**1**	0	1	0	0	**3**	**33.33**
	-Fe	0	0	**1**	0	2	0	**3**	**33.33**
	-Mg	1	0	0	**0**	0	0	**1**	**0.00**
	-N	3	0	0	3	**1**	1	**8**	**12.50**
	-K	3	0	0	2	2	**0**	**7**	**0.00**
	Total	**10**	**1**	**1**	**6**	**5**	**2**	**25**	
	Precision (%)	**20.00**	**100.00**	**100.00**	**0.00**	**20.00**	**0.00**		

B.16 Day 16

Table B.31: Confusion matrix of the first participant on day 16

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**0**	0	0	1	0	1	**2**	**0.00**
	-Ca	2	**1**	0	1	0	0	**4**	**25.00**
	-Fe	0	0	**4**	2	0	1	**7**	**57.14**
	-Mg	5	0	0	**0**	1	0	**6**	**0.00**
	-N	3	0	0	0	**1**	1	**5**	**20.00**
	-K	2	0	0	1	2	**0**	**5**	**0.00**
	Total	**12**	**1**	**4**	**5**	**4**	**3**	**29**	
	Precision (%)	**0.00**	**100.00**	**100.00**	**0.00**	**25.00**	**0.00**		

Table B.32: Confusion matrix of the second participant on day 16

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**0**	0	0	1	0	2	**3**	**0.00**
	-Ca	2	**3**	0	0	0	1	**6**	**50.00**
	-Fe	1	0	**1**	0	3	0	**5**	**20.00**
	-Mg	2	0	0	**0**	0	1	**3**	**0.00**
	-N	2	0	0	2	**0**	2	**6**	**0.00**
	-K	4	0	0	0	2	**1**	**7**	**14.29**
	Total	**11**	**3**	**1**	**3**	**5**	**7**	**30**	
	Precision (%)	**0.00**	**100.00**	**100.00**	**0.00**	**0.00**	**14.29**		

B.17 Day 17

Table B.33: Confusion matrix of the first participant on day 17

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	1	1	0	1	1	**8**	**50.00**
	-Ca	3	**2**	0	2	2	0	**9**	**22.22**
	-Fe	0	1	**1**	3	0	1	**6**	**16.67**
	-Mg	0	0	0	**3**	1	1	**5**	**60.00**
	-N	0	1	0	0	**0**	1	**2**	**0.00**
	-K	2	1	0	0	0	**1**	**4**	**25.00**
	Total	**9**	**6**	**2**	**8**	**4**	**5**	**34**	
	Precision (%)	**44.44**	**33.33**	**50.00**	**37.50**	**0.00**	**20.00**		

Table B.34: Confusion matrix of the second participant on day 17

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	0	0	0	0	2	**3**	**33.33**
	-Ca	0	**2**	0	0	0	0	**2**	**100.00**
	-Fe	1	0	**1**	0	0	0	**2**	**50.00**
	-Mg	4	0	0	**2**	0	0	**6**	**33.33**
	-N	3	1	0	0	**0**	0	**4**	**0.00**
	-K	5	0	0	0	1	**0**	**6**	**0.00**
	Total	**14**	**3**	**1**	**2**	**1**	**2**	**23**	
	Precision (%)	**7.14**	**66.67**	**100.00**	**100.00**	**0.00**	**0.00**		

B.18 Day 18

Table B.35: Confusion matrix of the first participant on day 18

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	0	3	0	**6**	**50.00**
	-Ca	0	**4**	0	0	0	0	**4**	**100.00**
	-Fe	2	1	**4**	1	1	1	**10**	**40.00**
	-Mg	2	1	0	**1**	0	1	**5**	**20.00**
	-N	2	0	0	0	**2**	1	**5**	**40.00**
	-K	4	1	0	1	0	**0**	**6**	**0.00**
	Total	**13**	**7**	**4**	**3**	**6**	**3**	**36**	
	Precision (%)	**23.08**	**57.14**	**100.00**	**33.33**	**33.33**	**0.00**		

Table B.36: Confusion matrix of the second participant on day 18

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	1	**5**	**80.00**
	-Ca	1	**2**	0	0	0	0	**3**	**66.67**
	-Fe	1	0	**1**	1	1	0	**4**	**25.00**
	-Mg	4	1	0	**1**	0	0	**6**	**16.67**
	-N	3	0	0	0	**3**	0	**6**	**50.00**
	-K	3	0	0	0	1	**1**	**5**	**20.00**
	Total	**16**	**3**	**1**	**2**	**5**	**2**	**29**	
	Precision (%)	**25.00**	**66.67**	**100.00**	**50.00**	**60.00**	**50.00**		

B.19 Day 19

Table B.37: Confusion matrix of the first participant on day 19

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	1	1	1	0	1	**8**	**50.00**
	-Ca	2	**4**	0	0	0	0	**6**	**66.67**
	-Fe	0	1	**2**	0	0	1	**4**	**50.00**
	-Mg	2	0	0	**0**	1	2	**5**	**0.00**
	-N	3	0	0	0	**1**	0	**4**	**25.00**
	-K	3	0	0	2	1	**1**	**7**	**14.29**
	Total	**14**	**6**	**3**	**3**	**3**	**5**	**34**	
	Precision (%)	**28.57**	**66.67**	**66.67**	**0.00**	**33.33**	**20.00**		

Table B.38: Confusion matrix of the second participant on day 19

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**5**	2	0	0	0	0	**7**	**71.43**
	-Ca	0	**2**	0	0	0	0	**2**	**100.00**
	-Fe	1	0	**1**	0	0	0	**2**	**50.00**
	-Mg	1	0	0	**1**	0	2	**4**	**25.00**
	-N	3	1	0	0	**1**	2	**7**	**14.29**
	-K	3	0	0	1	0	**3**	**7**	**42.86**
	Total	**13**	**5**	**1**	**2**	**1**	**7**	**29**	
	Precision (%)	**38.46**	**40.00**	**100.00**	**50.00**	**100.00**	**42.86**		

B.20 Day 21

Table B.39: Confusion matrix of the first participant on day 21

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**2**	2	0	0	1	1	**6**	**33.33**
	-Ca	0	**1**	0	0	0	0	**1**	**100.00**
	-Fe	0	1	**3**	0	1	1	**6**	**50.00**
	-Mg	0	1	0	**2**	2	1	**6**	**33.33**
	-N	0	1	0	0	**1**	0	**2**	**50.00**
	-K	1	0	0	1	1	**0**	**3**	**0.00**
	Total	**3**	**6**	**3**	**3**	**6**	**3**	**24**	
	Precision (%)	**66.67**	**16.67**	**100.00**	**66.67**	**16.67**	**0.00**		

Table B.40: Confusion matrix of the second participant on day 21

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	2	0	1	1	0	**5**	**20.00**
	-Ca	2	**3**	0	0	0	0	**5**	**60.00**
	-Fe	1	0	**1**	0	0	1	**3**	**33.33**
	-Mg	2	0	0	**3**	1	0	**6**	**50.00**
	-N	3	1	0	0	**1**	0	**5**	**20.00**
	-K	4	0	0	1	0	**0**	**5**	**0.00**
	Total	**13**	**6**	**1**	**5**	**3**	**1**	**29**	
	Precision (%)	**7.69**	**50.00**	**100.00**	**60.00**	**33.33**	**0.00**		

B.21 Day 22

Table B.41: Confusion matrix of the first participant on day 22

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	2	0	0	0	0	**6**	**66.67**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	1	**4**	0	1	0	**6**	**66.67**
	-Mg	0	1	0	**3**	0	1	**5**	**60.00**
	-N	2	0	0	0	**0**	1	**3**	**0.00**
	-K	2	0	0	0	0	**0**	**2**	**0.00**
	Total	**9**	**4**	**4**	**3**	**1**	**2**	**23**	
	Precision (%)	**44.44**	**0.00**	**100.00**	**100.00**	**0.00**	**0.00**		

Table B.42: Confusion matrix of the second participant on day 22

Actual \ Predicted	Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Healthy	**4**	1	0	0	0	1	**6**	**66.67**
-Ca	1	**0**	0	0	0	0	**1**	**0.00**
-Fe	0	0	**1**	0	3	2	**6**	**16.67**
-Mg	0	2	0	**3**	0	0	**5**	**60.00**
-N	6	1	0	0	**1**	0	**8**	**12.50**
-K	6	0	0	0	0	**1**	**7**	**14.29**
Total	**17**	**4**	**1**	**3**	**4**	**4**	**33**	
Precision (%)	**23.53**	**0.00**	**100.00**	**100.00**	**25.00**	**25.00**		

B.22 Day 23

Table B.43: Confusion matrix of the first participant on day 23

Actual \ Predicted	Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Healthy	**3**	0	0	0	1	0	**4**	**75.00**
-Ca	0	**0**	0	0	1	0	**1**	**0.00**
-Fe	1	1	**4**	0	2	1	**9**	**44.44**
-Mg	1	1	0	**6**	1	0	**9**	**66.67**
-N	1	0	0	0	**1**	1	**3**	**33.33**
-K	2	1	0	0	1	**3**	**7**	**42.86**
Total	**8**	**3**	**4**	**6**	**7**	**5**	**33**	
Precision (%)	**37.50**	**0.00**	**100.00**	**100.00**	**14.29**	**60.00**		

Table B.44: Confusion matrix of the second participant on day 23

Actual \ Predicted	Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Healthy	**2**	0	0	0	0	0	**2**	**100.00**
-Ca	0	**0**	0	0	0	1	**1**	**0.00**
-Fe	0	0	**0**	0	1	1	**2**	**0.00**
-Mg	0	1	0	**3**	0	1	**5**	**60.00**
-N	2	1	0	1	**2**	1	**7**	**28.57**
-K	2	0	0	0	0	**0**	**2**	**0.00**
Total	**6**	**2**	**0**	**4**	**3**	**4**	**19**	
Precision (%)	**33.33**	**0.00**	**0.00**	**75.00**	**66.67**	**0.00**		

B.23 Day 24

Table B.45: Confusion matrix of the first participant on day 24

Actual \ Predicted	Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Healthy	**0**	2	0	0	2	1	**5**	**0.00**
-Ca	1	**1**	0	0	0	0	**2**	**50.00**
-Fe	1	0	**1**	2	1	0	**5**	**20.00**
-Mg	0	0	0	**2**	1	2	**5**	**40.00**
-N	0	1	0	0	**2**	1	**4**	**50.00**
-K	2	0	0	0	5	**0**	**7**	**0.00**
Total	**4**	**4**	**1**	**4**	**11**	**4**	**28**	
Precision (%)	**0.00**	**25.00**	**100.00**	**50.00**	**18.18**	**0.00**		

Table B.46: Confusion matrix of the second participant on day 24

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	4	0	0	1	1	**7**	**14.29**
	-Ca	2	**1**	0	0	0	0	**3**	**33.33**
	-Fe	1	0	**2**	0	1	2	**6**	**33.33**
	-Mg	1	2	0	**2**	1	0	**6**	**33.33**
	-N	1	1	0	0	**3**	0	**5**	**60.00**
	-K	2	1	0	0	5	**1**	**9**	**11.11**
	Total	**8**	**9**	**2**	**2**	**11**	**4**	**36**	
	Precision (%)	**12.50**	**11.11**	**100.00**	**100.00**	**27.27**	**25.00**		

B.24 Day 25

Table B.47: Confusion matrix of the first participant on day 25

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	0	0	0	0	2	**3**	**33.33**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	0	0	**3**	0	0	0	**3**	**100.00**
	-Mg	1	1	0	**0**	0	0	**2**	**0.00**
	-N	1	2	3	0	**0**	0	**6**	**0.00**
	-K	1	2	0	0	0	**3**	**6**	**50.00**
	Total	**4**	**5**	**6**	**0**	**0**	**5**	**20**	
	Precision (%)	**25.00**	**0.00**	**50.00**	**0.00**	**0.00**	**60.00**		

Table B.48: Confusion matrix of the second participant on day 25

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	2	0	0	0	1	**4**	**25.00**
	-Ca	1	**1**	0	0	0	0	**2**	**50.00**
	-Fe	0	0	**3**	0	0	1	**4**	**75.00**
	-Mg	1	0	0	**2**	1	2	**6**	**33.33**
	-N	1	4	0	0	**0**	1	**6**	**0.00**
	-K	2	0	0	0	1	**0**	**3**	**0.00**
	Total	**6**	**7**	**3**	**2**	**2**	**5**	**25**	
	Precision (%)	**16.67**	**14.29**	**100.00**	**100.00**	**0.00**	**0.00**		

B.25 Day 26

Table B.49: Confusion matrix of the first participant on day 26

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	1	0	0	1	1	**4**	**25.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	1	0	**1**	0	0	0	**2**	**50.00**
	-Mg	0	2	0	**2**	0	0	**4**	**50.00**
	-N	0	1	0	0	**1**	0	**2**	**50.00**
	-K	2	0	0	0	0	**3**	**5**	**60.00**
	Total	**5**	**4**	**1**	**2**	**2**	**4**	**18**	
	Precision (%)	**20.00**	**0.00**	**100.00**	**100.00**	**50.00**	**75.00**		

Table B.50: Confusion matrix of the second participant on day 26

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**0**	1	0	0	0	0	**1**	**0.00**
	-Ca	0	**1**	0	0	0	0	**1**	**100.00**
	-Fe	2	0	**0**	0	0	1	**3**	**0.00**
	-Mg	0	3	0	**1**	1	0	**5**	**20.00**
	-N	3	2	0	1	**0**	1	**7**	**0.00**
	-K	0	0	0	1	1	**1**	**3**	**33.33**
	Total	**5**	**7**	**0**	**3**	**2**	**3**	**20**	
	Precision (%)	**0.00**	**14.29**	**0.00**	**33.33**	**0.00**	**33.33**		

B.26 Day 27

Table B.51: Confusion matrix of the first participant on day 27

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**0**	4	0	0	1	0	**5**	**0.00**
	-Ca	0	**1**	0	0	0	0	**1**	**100.00**
	-Fe	0	2	**1**	0	1	0	**4**	**25.00**
	-Mg	0	0	0	**0**	1	2	**3**	**0.00**
	-N	0	1	1	0	**0**	1	**3**	**0.00**
	-K	3	0	0	0	1	**2**	**6**	**33.33**
	Total	**3**	**8**	**2**	**0**	**4**	**5**	**22**	
	Precision (%)	**0.00**	**12.50**	**50.00**	**0.00**	**0.00**	**40.00**		

Table B.52: Confusion matrix of the second participant on day 27

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**0**	3	0	0	0	1	**4**	**0.00**
	-Ca	2	**1**	0	0	0	0	**3**	**33.33**
	-Fe	1	0	**0**	1	0	1	**3**	**0.00**
	-Mg	0	0	0	**2**	0	1	**3**	**66.67**
	-N	3	2	1	0	**0**	1	**7**	**0.00**
	-K	3	0	0	0	2	**1**	**6**	**16.67**
	Total	**9**	**6**	**1**	**3**	**2**	**5**	**26**	
	Precision (%)	**0.00**	**16.67**	**0.00**	**66.67**	**0.00**	**20.00**		

B.27 Day 28

Table B.53: Confusion matrix of the first participant on day 28

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	0	0	0	0	2	**3**	**33.33**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	2	**0**	0	1	0	**3**	**0.00**
	-Mg	0	2	0	**0**	0	0	**2**	**0.00**
	-N	0	1	0	0	**1**	2	**4**	**25.00**
	-K	3	0	1	0	0	**2**	**6**	**33.33**
	Total	**5**	**5**	**1**	**0**	**2**	**6**	**19**	
	Precision (%)	**20.00**	**0.00**	**0.00**	**0.00**	**50.00**	**33.33**		

Table B.54: Confusion matrix of the second participant on day 28

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	2	0	0	0	1	**4**	**25.00**
	-Ca	0	**2**	0	0	0	0	**2**	**100.00**
	-Fe	0	1	**0**	0	0	2	**3**	**0.00**
	-Mg	0	0	0	**2**	0	0	**2**	**100.00**
	-N	0	1	0	1	**1**	1	**4**	**25.00**
	-K	6	1	0	0	1	**1**	**9**	**11.11**
Total		**7**	**7**	**0**	**3**	**2**	**5**	**24**	
Precision (%)		**14.29**	**28.57**	**0.00**	**66.67**	**50.00**	**20.00**		

Appendix C: System classification performance

The confusion matrix of the system are separate in the 28 days in Sections C.1 to C.28.

C.1 Day 1

Table C.1: Confusion matrix of the system on day 1

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	0	0	0	**3**	**100.00**
	-Ca	2	**0**	0	0	0	0	**2**	**0.00**
	-Fe	0	0	**0**	0	0	0	**0**	**0.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	1	0	0	0	**0**	0	**1**	**0.00**
	-K	1	0	0	0	0	**0**	**1**	**0.00**
	Total	**7**	**0**	**0**	**0**	**0**	**0**	**7**	
	Precision (%)	**42.86**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.2 Day 2

Table C.2: Confusion matrix of the system on day 2

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	0	0	**0**	0	0	0	**0**	**0.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	0	0	0	0	**0**	0	**0**	**0.00**
	-K	0	0	0	0	0	**0**	**0**	**0.00**
	Total	**4**	**0**	**0**	**0**	**0**	**0**	**4**	
	Precision (%)	**100.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.3 Day 3

Table C.3: Confusion matrix of the system on day 3

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	4	0	0	0	0	0	4	100.00
	-Ca	1	0	0	0	0	0	1	0.00
	-Fe	0	0	0	0	0	0	0	0.00
	-Mg	0	0	0	0	0	0	0	0.00
	-N	0	0	0	0	0	0	0	0.00
	-K	0	0	0	0	0	0	0	0.00
	Total	5	0	0	0	0	0	5	
	Precision (%)	80.00	0.00	0.00	0.00	0.00	0.00		

C.4 Day 4

Table C.4: Confusion matrix of the system on day 4

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	1	0	**0**	0	1	0	**2**	**0.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	3	0	0	0	**0**	0	**3**	**0.00**
	-K	0	0	0	1	0	**0**	**1**	**0.00**
	Total	**9**	**0**	**0**	**1**	**1**	**0**	**11**	
	Precision (%)	**44.44**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.5 Day 5

Table C.5: Confusion matrix of the system on day 5

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	0	0	**0**	0	0	0	**0**	**0.00**
	-Mg	1	0	0	**0**	0	0	**1**	**0.00**
	-N	0	0	0	0	**0**	0	**0**	**0.00**
	-K	0	0	0	0	0	**0**	**0**	**0.00**
	Total	**5**	**0**	**0**	**0**	**0**	**0**	**5**	
	Precision (%)	**80.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.6 Day 6

Table C.6: Confusion matrix of the system on day 6

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	0	**0**	1	0	0	**1**	**0.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	0	0	0	0	**0**	0	**0**	**0.00**
	-K	1	0	0	0	0	**0**	**1**	**0.00**
	Total	**6**	**0**	**0**	**1**	**0**	**0**	**7**	
	Precision (%)	**66.67**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.7 Day 7

Table C.7: Confusion matrix of the system on day 7

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	0	**0**	0	0	0	**0**	**0.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	0	0	0	0	**0**	0	**0**	**0.00**
	-K	1	0	0	0	0	**0**	**1**	**0.00**
	Total	**6**	**0**	**0**	**0**	**0**	**0**	**6**	
	Precision (%)	**66.67**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.8 Day 8

Table C.8: Confusion matrix of the system on day 8

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	0	0	0	**3**	**100.00**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	0	0	**0**	0	0	0	**0**	**0.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	0	0	0	**0**	**0**	**0.00**
	Total	**3**	**0**	**0**	**1**	**0**	**0**	**4**	
	Precision (%)	**100.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.9 Day 9

Table C.9: Confusion matrix of the system on day 9

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	1	0	**0**	1	1	0	**3**	**0.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	0	0	0	**0**	**0**	**0.00**
	Total	**5**	**0**	**0**	**2**	**1**	**0**	**8**	
	Precision (%)	**80.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		

C.10 Day 10

Table C.10: Confusion matrix of the system on day 10

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	2	**0**	1	0	1	0	**4**	**0.00**
	-Fe	0	0	**0**	1	2	0	**3**	**0.00**
	-Mg	2	0	0	**0**	1	0	**3**	**0.00**
	-N	2	0	0	0	**1**	0	**3**	**33.33**
	-K	1	0	0	0	1	**0**	**2**	**0.00**
	Total	**11**	**0**	**1**	**1**	**6**	**0**	**19**	
	Precision (%)	**36.36**	**0.00**	**0.00**	**0.00**	**16.67**	**0.00**		

C.11 Day 11

Table C.11: Confusion matrix of the system on day 11

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	2	0	0	**2**	**0.00**
	-Fe	0	0	**2**	0	1	0	**3**	**66.67**
	-Mg	0	0	1	**0**	0	0	**1**	**0.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	1	0	0	**0**	**1**	**0.00**
	Total	**4**	**0**	**4**	**3**	**1**	**0**	**12**	
	Precision (%)	**100.00**	**0.00**	**50.00**	**0.00**	**0.00**	**0.00**		

C.12 Day 12

Table C.12: Confusion matrix of the system on day 12

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	0	0	**2**	0	0	0	**2**	**100.00**
	-Mg	0	0	0	**0**	0	0	**0**	**0.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	0	0	0	**0**	**0**	**0.00**
	Total	**4**	**0**	**2**	**1**	**0**	**0**	**7**	
	Precision (%)	**100.00**	**0.00**	**100.00**	**0.00**	**0.00**	**0.00**		

C.13 Day 13

Table C.13: Confusion matrix of the system on day 13

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	0	0	**2**	0	0	0	**2**	**100.00**
	-Mg	0	0	0	**1**	0	0	**1**	**100.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	1	0	0	**0**	**1**	**0.00**
	Total	**4**	**0**	**3**	**2**	**0**	**0**	**9**	
	Precision (%)	**100.00**	**0.00**	**66.67**	**50.00**	**0.00**	**0.00**		

C.14 Day 14

Table C.14: Confusion matrix of the system on day 14

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	1	0	0	**1**	**0.00**
	-Fe	0	0	**2**	0	0	0	**2**	**100.00**
	-Mg	0	0	0	**1**	0	0	**1**	**100.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	0	0	1	**0**	**1**	**0.00**
	Total	**4**	**0**	**2**	**3**	**1**	**0**	**10**	
	Precision (%)	**100.00**	**0.00**	**100.00**	**33.33**	**0.00**	**0.00**		

C.15 Day 15

Table C.15: Confusion matrix of the system on day 15

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	1	0	0	**1**	**0.00**
	-Fe	0	0	**2**	1	0	0	**3**	**66.67**
	-Mg	0	0	0	**1**	0	0	**1**	**100.00**
	-N	0	0	0	1	**1**	0	**2**	**50.00**
	-K	0	0	0	1	0	**0**	**1**	**0.00**
	Total	**4**	**0**	**2**	**5**	**1**	**0**	**12**	
	Precision (%)	**100.00**	**0.00**	**100.00**	**20.00**	**100.00**	**0.00**		

C.16 Day 16

Table C.16: Confusion matrix of the system on day 16

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	0	**2**	0	0	0	**2**	**100.00**
	-Mg	0	0	0	**1**	0	0	**1**	**100.00**
	-N	1	0	0	0	**0**	0	**1**	**0.00**
	-K	0	0	1	0	0	**0**	**1**	**0.00**
	Total	**6**	**0**	**3**	**1**	**0**	**0**	**10**	
	Precision (%)	**66.67**	**0.00**	**66.67**	**100.00**	**0.00**	**0.00**		

C.17 Day 17

Table C.17: Confusion matrix of the system on day 17

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	4	0	0	0	0	0	**4**	**100.00**
	-Ca	2	**0**	0	0	0	0	**2**	**0.00**
	-Fe	0	0	**2**	0	0	0	**2**	**100.00**
	-Mg	0	0	0	**1**	0	0	**1**	**100.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	0	0	0	**0**	**0**	**0.00**
	Total	**6**	**0**	**2**	**2**	**0**	**0**	**10**	
	Precision (%)	**66.67**	**0.00**	**100.00**	**50.00**	**0.00**	**0.00**		

C.18 Day 18

Table C.18: Confusion matrix of the system on day 18

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	0	1	0	**4**	**75.00**
	-Ca	0	**0**	0	1	0	0	**1**	**0.00**
	-Fe	0	0	**2**	1	0	0	**3**	**66.67**
	-Mg	0	0	0	**1**	0	0	**1**	**100.00**
	-N	0	0	1	3	**0**	0	**4**	**0.00**
	-K	1	0	2	0	0	**0**	**3**	**0.00**
	Total	**4**	**0**	**5**	**6**	**1**	**0**	**16**	
	Precision (%)	**75.00**	**0.00**	**40.00**	**16.67**	**0.00**	**0.00**		

C.19 Day 19

Table C.19: Confusion matrix of the system on day 19

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	0	**0**	0	0	0	0	**0**	**0.00**
	-Fe	0	0	**2**	1	0	0	**3**	**66.67**
	-Mg	0	0	0	**2**	0	0	**2**	**100.00**
	-N	1	0	0	1	**0**	0	**2**	**0.00**
	-K	0	0	0	1	0	**0**	**1**	**0.00**
	Total	**5**	**0**	**2**	**5**	**0**	**0**	12	
	Precision (%)	**80.00**	**0.00**	**100.00**	**40.00**	**0.00**	**0.00**		

C.20 Day 21

Table C.20: Confusion matrix of the system on day 21

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	1	**0**	0	1	0	0	**2**	**0.00**
	-Fe	0	0	**2**	2	0	0	**4**	**50.00**
	-Mg	0	0	0	**2**	0	0	**2**	**100.00**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	0	2	0	**0**	**2**	**0.00**
	Total	**5**	**0**	**2**	**8**	**0**	**0**	**15**	
	Precision (%)	**80.00**	**0.00**	**100.00**	**25.00**	**0.00**	**0.00**		

C.21 Day 22

Table C.21: Confusion matrix of the system on day 22

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**4**	0	0	0	0	0	**4**	**100.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	0	**3**	1	0	0	**4**	**75.00**
	-Mg	0	0	0	**1**	0	0	**1**	**100.00**
	-N	0	0	0	3	**0**	0	**3**	**0.00**
	-K	0	0	1	1	0	**0**	**2**	**0.00**
Total		**5**	**0**	**4**	**6**	**0**	**0**	**15**	
Precision (%)		**80.00**	**0.00**	**75.00**	**16.67**	**0.00**	**0.00**		

C.22 Day 23

Table C.22: Confusion matrix of the system on day 23

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	0	1	0	**4**	**75.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	0	**4**	0	0	0	**4**	**100.00**
	-Mg	0	0	0	**2**	0	0	**2**	**100.00**
	-N	1	0	2	0	**0**	0	**3**	**0.00**
	-K	0	0	2	0	0	**0**	**2**	**0.00**
	Total	**5**	**0**	**8**	**2**	**1**	**0**	**16**	
	Precision (%)	**60.00**	**0.00**	**50.00**	**100.00**	**0.00**	**0.00**		

C.23 Day 24

Table C.23: Confusion matrix of the system on day 24

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	0	2	0	1	0	**4**	**25.00**
	-Ca	1	**0**	0	0	0	0	**1**	**0.00**
	-Fe	0	0	**3**	1	0	0	**4**	**75.00**
	-Mg	1	0	0	**1**	1	0	**3**	**33.33**
	-N	0	0	1	2	**0**	0	**3**	**0.00**
	-K	0	0	1	1	2	**0**	**4**	**0.00**
	Total	**3**	**0**	**7**	**5**	**4**	**0**	**19**	
	Precision (%)	**33.33**	**0.00**	**42.86**	**20.00**	**0.00**	**0.00**		

C.24 Day 25

Table C.24: Confusion matrix of the system on day 25

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**2**	0	0	1	1	0	**4**	**50.00**
	-Ca	1	**0**	0	1	0	0	**2**	**0.00**
	-Fe	0	0	**3**	1	0	0	**4**	**75.00**
	-Mg	1	0	0	**1**	0	0	**2**	**50.00**
	-N	0	0	1	0	**2**	0	**3**	**66.67**
	-K	0	0	0	3	0	**0**	**3**	**0.00**
	Total	**4**	**0**	**4**	**7**	**3**	**0**	**18**	
	Precision (%)	**50.00**	**0.00**	**75.00**	**14.29**	**66.67**	**0.00**		

C.25 Day 26

Table C.25: Confusion matrix of the system on day 26

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	1	0	0	0	**4**	**75.00**
	-Ca	0	**0**	0	0	1	0	**1**	**0.00**
	-Fe	0	0	**1**	3	0	0	**4**	**25.00**
	-Mg	0	0	0	**1**	1	0	**2**	**50.00**
	-N	0	0	0	3	**0**	0	**3**	**0.00**
	-K	0	0	2	0	1	**0**	**3**	**0.00**
	Total	**3**	**0**	**4**	**7**	**3**	**0**	17	
	Precision (%)	**100.00**	**0.00**	**25.00**	**14.29**	**0.00**	**0.00**		

C.26 Day 27

Table C.26: Confusion matrix of the system on day 27

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**1**	0	1	1	1	0	**4**	**25.00**
	-Ca	0	**0**	0	1	0	0	**1**	**0.00**
	-Fe	0	0	**1**	3	0	0	**4**	**25.00**
	-Mg	0	0	0	**2**	0	0	**2**	**100.00**
	-N	0	0	0	2	**0**	0	**2**	**0.00**
	-K	0	0	1	1	0	**0**	**2**	**0.00**
Total		**1**	**0**	**3**	**10**	**1**	**0**	15	
Precision (%)		**100.00**	**0.00**	**33.33**	**20.00**	**0.00**	**0.00**		

C.27 Day 28

Table C.27: Confusion matrix of the system on day 28

		Predicted							
		Healthy	-Ca	-Fe	-Mg	-N	-K	**Total**	**Recall (%)**
Actual	Healthy	**3**	0	0	1	0	0	**4**	**75.00**
	-Ca	0	**0**	0	3	0	0	**3**	**0.00**
	-Fe	1	0	**2**	0	0	0	**3**	**66.67**
	-Mg	1	0	0	**2**	0	0	**3**	**66.67**
	-N	0	0	0	1	**0**	0	**1**	**0.00**
	-K	0	0	0	1	1	**0**	**2**	**0.00**
	Total	**5**	**0**	**2**	**8**	**1**	**0**	**16**	
	Precision (%)	**60.00**	**0.00**	**100.00**	**25.00**	**0.00**	**0.00**		