



**CLASSIFYING SALEABILITY OF LIME BY USING
CONVOLUTIONAL NEURAL NETWORK (CNN) APPROACH**

BY

NIRACHA CHAIWONG

**AN INDEPENDENT STUDY SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF ENGINEERING (LOGISTICS AND SUPPLY
CHAIN SYSTEMS ENGINEERING)
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2021
COPYRIGHT OF THAMMASAT UNIVERSITY**

THAMMASAT UNIVERSITY
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

INDEPENDENT STUDY

BY

NIRACHA CHAIWONG


ENTITLED

CLASSIFYING SALEABILITY OF LIME BY USING CONVOLUTIONAL NEURAL
NETWORK (CNN) APPROACH

was approved as partial fulfillment of the requirements for
the degree of Master of Engineering (Logistics and Supply Chain Systems Engineering)

on June 13, 2022

Member and Advisor



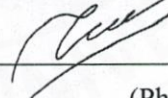
(Assistant Professor Warut Pannakkong, Ph.D.)

Member and Co-advisor



(Associate Professor Jirachai Buddhakulsomsiri, Ph.D.)

Member



(Pham Duc Tai, Ph.D.)

Director



(Professor Pruettha Nanakorn, D.Eng.)

Independent Study Title	CLASSIFYING SALEABILITY OF LIME BY USING CONVOLUTIONAL NEURAL NETWORK (CNN) APPROACH
Author	Niracha Chaiwong
Degree	Master of Engineering (Logistics and Supply Chain Systems Engineering)
Faculty/University	Sirindhorn International Institute of Technology/ Thammasat University
Advisor	Assistant Professor Warut Pannakkong, Ph.D
Co-Advisor	Associate Professor Jirachai Buddhakulsomsiri, Ph.D.
Academic Years	2021

ABSTRACT

Nowadays, there are many applications that have been used in agriculture field to facilitate human workers. Many researches have been focused on fruits grading based on appearance features. Since lime is one of Thai economic crop that have high demand and price all the year, a lot of limes are continually exported into the market. The appearance of limes strongly impact the market value. So, the quality inspection is needed to avoid selling bad limes to customers which cause greatly affects to the customer base. Determining saleability of lime can be done by using humans or machines. Using human workers is time-consuming, high cost and not accurate. Although there are fruit grading machines, they are expensive and difficult to reach for general farmers. This research has proposed method to classify saleability of lime into three categories which are buy, maybe, and not buy by using Convolutional Neural Network (CNN) because it is a deep neural network that has high performance in terms of images classification tasks. Hyperparameters tuning process has been used to search for the best model performance by using full factorial design. The hyperparameters that have been considered in the model are Epoch, Learning Rate, Decaying Rate and Momentum. Two experiments of hyperparameters tuning were constructed. The first experiment constructed by initial 2^5 full factorial by setting low and high value. After

that, the result from first experiment was analysed by using Minitab to remove unimportant factors. The model performance in this research has been evaluated by F1-score which the first experiment gave the highest results as 76%, 79%, 80% in training model, 73%, 77%, 77% in validation and 74%, 77%, 77% of testing for buy, maybe and not buy respectively.

Keywords: Neural Network, Convolutional Neureal Network (CNN), Artificial Intelligent, Lime prediction, Saleabilty of lime



ACKNOWLEDGEMENTS

I would like to express my special thanks to my advisor, Assistant Professor Dr. Warut Pannakkong, Associate Professor Dr. Jirachai Buddhakulsomsiri and Dr. Pham Duc Tai who are invaluable supportive for this project, gave advice and constant encouragement. I strongly appreciate a Lime farm at Ratchaburi, Thailand which provided information and lime samples for this project. Moreover, Sirindhorn International Institute of Technology (SIIT), Thammasat University which provided facility area so that I was able to conduct the experiment and research. I am also grateful to my family who encourage me from the beginning throughout this project. And finally, this work would not be possible without the funding and support from SIIT.

Niracha Chaiwong

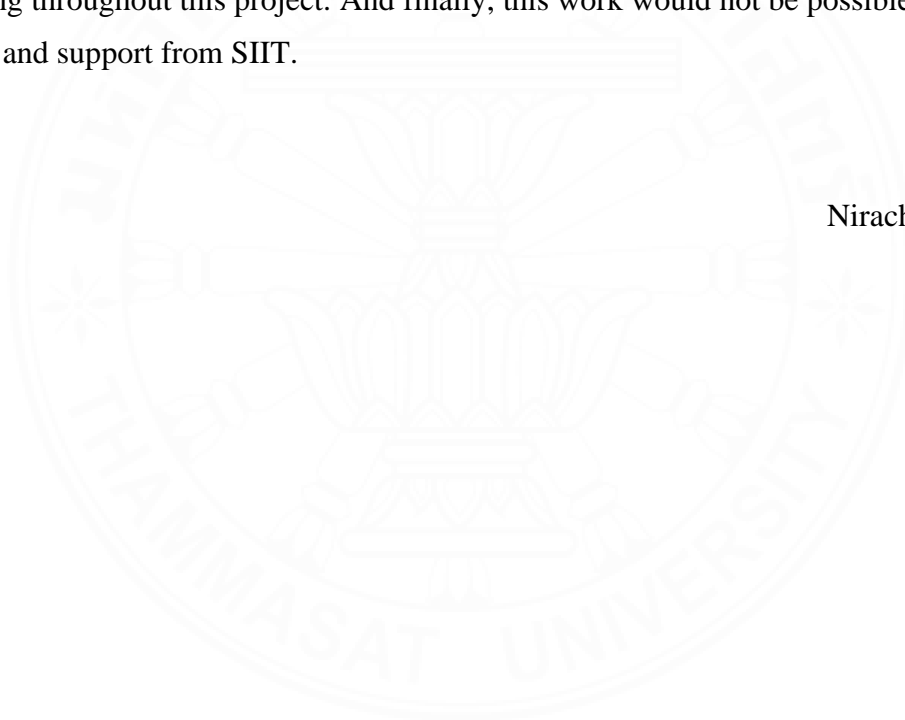


TABLE OF CONTENTS

	Page
ABSTRACT	(1)
ACKNOWLEDGEMENTS	(3)
LIST OF TABLES	(6)
LIST OF FIGURES	(7)
LIST OF SYMBOLS/ABBREVIATIONS	(8)
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement	2
1.2 Research Objective	2
CHAPTER 2 LITERATURE REVIEW	3
2.1 Related Work	3
2.2 Research Gap	5
2.3 Convolutional Neural Network (CNN)	6
2.3.1 Structure of Convolutional Neural Network (CNN)	6
2.3.2 Performance Evaluation	9
CHAPTER 3 METHODOLOGY	10
3.1 Data Collection	11
3.1.1 Image Acquisition	11
3.1.2 Visual Inspection	12
3.2 Data Preparation	13
3.3 Convolutional Neural Network (CNN) Model	13
3.3.1 Structure of Convolutional Neural Network (CNN) Model	13
3.3.2 Process of Convolutional Neural Network (CNN)	14
3.3.3 Model Selection	15

	(5)
3.4 Hyperparameters Tuning	16
3.4.1 Initial Full Factorial Design	16
3.4.2 Reduced Full Factorial Design	18
CHAPTER 4 RESULT AND DISCUSSION	19
4.1 Result of Hyperparameters Tuning Process	19
4.1.1 Result of Initial Full Factorial Design	19
4.1.2 Result After Reduced Full Factorial Design	20
4.1.3 Result Comparison	21
4.2 Discussion	24
CHAPTER 5 CONCLUSION AND FUTURE WORK	25
5.1 Conclusion	25
5.2 Future Work	25
REFERENCES	26
APPENDICES	29
APPENDIX A	30
APPENDIX B	32
APPENDIX C	38
BIOGRAPHY	48

LIST OF TABLES

Tables	Page
2.1 Summary of related research.	5
3.1 Visual inspection table.	12
3.2 2^5 full factorial design low and high value setting.	16
3.3 Hyperparameters setting table in initial full factorial design.	17
4.1 Summary result for training set in initial full factorial design.	19
4.2 Summary result for validation set in initial full factorial design.	20
4.3 Summary result for testing set in initial full factorial design.	20
4.4 Summary result for training set after reduced full factorial design.	20
4.5 Summary result for validation set after reduced full factorial design.	21
4.6 Summary result for testing set after reduced full factorial design.	21
4.7 Best hyperparameters setting.	23
4.8 Summary of the best F1-score result.	23
A.1 First experiment for hyperparameter setting.	30
A.2 Second experiment for hyperparameter setting.	31
B.1 Training result from first experiment.	32
B.2 Validation result from first experiment.	33
B.3 Testing result from first experiment.	34
B.4 Training result from second experiment.	35
B.5 Validation result from second experiment.	36
B.6 Testing result from second experiment.	37

LIST OF FIGURES

Figures	Page
2.1 Layers of CNN model (Eremenko, 2018).	6
2.2 Convolutional layer (Eremenko, 2018).	7
2.3 Pooling layer (Eremenko, 2018).	7
2.4 Flattening layer (Eremenko, 2018).	8
2.5 Fully connected layer (Eremenko, 2018).	8
3.1 Methodology process.	10
3.2 Equipment setup.	11
3.3 Four angles of lime image.	12
3.4 Structure of CNN model (Sudha & Aji, 2021).	13
3.5 Training process of model.	15
3.6 Hyperparameters tuning process.	18
4.1 Weighted average of F1-score in first experiment.	22
4.2 Weighted average of F1-score in second experiment.	22

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
KNN	K-Nearest Neighbor
SVM	Support Vector Machine
SVMR	Support Vector Machine Regression
FCM	Fuzzy C-Mean
RBF	Radial Basis Function
ANFIS	Clustered Adaptive Neuro-Fuzzy Inference System
SC	Subtractive Clustering
GP	Grid Partitioning
CVS	Computer Vision System
BPNN	Backpropagation Neural Network
CFS	Correlation-Based Feature Selection Subset
CONS	Consistency Subsets
RMSE	Root Mean Square Error
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

CHAPTER 1

INTRODUCTION

Citrus aurantifolia has been grown worldwide, particularly in the tropical and subtropical area. Citrus fruit is beneficial to people's health because it contains numerous natural metabolites. It can prevent heart, liver, bone, and urinary diseases (Narang & Jiraungkoorskul, 2016). Lime (Citrus aurantifolia Swingle cv. Paan) is an economic crop in Thailand which harvest season is from July to September. It is normally harvested when it has green skin and strong aromatic smell. The price of lime fruit is mostly determined by season and its quality (Kaewsuksaeng, Tatmala, Srilaong, & Pongprasert, 2015). Lime is used in a variety of Thai dishes and beverages. The selling price of lime fruit is constantly high which is attractive to agriculturists (Booranawong & Booranawong, 2018). Many parts of Thailand are suitable for planting lime which are central, southern, and northern (Booranawong & Booranawong, 2017).

In the agriculture field, automation has been used to improve country's quality, economic growth and productivity. Sorting fruits and vegetables have impact on export market which quality evaluation is needed. The appearance of the fruits and vegetables also strongly impact the market value, customer preference and decision. An autonomous system has the ability to reduce time, cost and human error which human workers are easily disrupted by the environment (Bhargava & Bansal, 2021). Before setting prices for fruits and vegetables, quality determination and product testing are needed for manufacturing process. In some food industries image processing techniques have been used to monitor quality of agricultural products (Pathmanaban, Gnanavel, & Anandan, 2019).

Academics and industries have been interested in agriculture post-harvest operation to reduce the losses (Phate, Malmathanraj, & Palanisamy, 2021). Computer vision has been improved to replace manual inspection and assist human workers to improve their skills which are classification, quality estimation by observing internal and external features (Saldaña, Siche, Luján, & Quevedo, 2013). Machine learning approaches were combined with sufficient image processing that have the potential to develop an automation system because fruit identification, classification and grading cannot be done accurately by human perception (Behera, Rath, Mahapatra, & Sethy, 2020).

1.1 Problem Statement

Lime is one of the economic crops in Thailand which has high demand all the year. Selling bad limes to customers can cause losing customers base and trust of company. So, the quality inspection process is needed to determine saleability of limes before selling in the market. The lime with green skin and fewer defects has more chance to sale than the defective lime. Although quality inspection can be done by human workers, using human is time consuming, high cost, and not accurate. Moreover, using machine is expensive and difficult to reach for general farmers. So, this research has proposed method to classify saleability of limes automatically by using Convolutional Neural Network (CNN).

1.2 Research Objective

The main objective of this research is to train the CNN model for classifying various quality of lime images into three categories which are buy, maybe and not buy. Furthermore, to evaluate the model performance in train, validation and test set. Lastly, to implement CNN model with real-world data set or test set of lime images that the model never seen before.

CHAPTER 2

LITERATURE REVIEW

2.1 Related Work

Computer vision and machine learning have been used in many applications in the agriculture field. This chapter is to review previous works which related to this paper. Clustered adaptive neuro-fuzzy inference system (ANFIS) has been developed to predict the weight of Indian sweet lime with 1D and 2D features which extracted from computer vision. Three clustering methods have been compared which are Fuzzy C-mean clustering (FCM), Subtractive clustering (SC) and Grid partitioning (GP). The result shows that using ANFIS with FCM gives the most accuracy and less error between 5.13% and -6.25% (Phate, Malmathanraj, & Palanisamy, 2019). The computer vision system (CVS) has been developed to determine sweet lime weight by using image processing to extract geometrical features. The dimensional features are used to predict the weight of the sweet lime. A support vector machine regression (SVMR) has been used to estimate the weight of the samples. The R^2 coefficient is 0.9866 and RMSE is 6.435 (Phate, Malmathanraj, & Palanisamy, 2020). Machine learning and Meta-heuristic approach have been developed for estimating sweet lime weight by using computer vision with GA-ANFIS and PSO-ANFIS which GA-ANFIS gave better result and less time with RMSE = 4.1581 (Phate et al., 2021). The defective and ripeness of tomatoes has been estimated by RGB extraction and ANN for classification with an accuracy of 96.47 % (Arakeri, 2016). Backpropagation neural network (BPNN) has been developed for tomato maturity detection which are green, orange and red. The RGB values are extracted from images and converted to HSI model. The results shows that H had the most accuracy to extract and identify the maturity of tomatoes by 99.31% with SD 1.2% (Wan, Toudeshki, Tan, & Ehsani, 2018). Mass and volume of cherry tomato has been predicted by support vector machine (SVM), radial basis function (RBF) and Bayesian artificial neural network (Bayesian-ANN) with 2D and 3D images analysis (Nyalala et al., 2019). Discriminate defects and grading of tomatoes by using color, texture and shape features to determine defects or healthy area extracted in LAB space. Histogram of the image corresponding to different grading which healthy tomato has higher pixel value than detected one. RBF-SVM was the best model which gave the most accuracy which estimated 0.9515 (Irer, Belal, Okinda, Makange, & Ji, 2019). SVM and K-means cluster

have been developed for classifying severity disease of orange with 90% accuracy (Behera, Jena, Rath, & Sethy, 2018). A three-variety automatic and non-instructive computer vision system has been developed to estimate PH value of orange based on hybrid ANN-ABC which can use with various orange types (ASabzi, Javadikia, & Arribas, 2020). A mobile platform also developed for pre-grading automation which can analyze color and size of citrus by using RGB extraction and sum of pixels from the images. R^2 coefficient of size is 0.993 and 0.918 for color (Cubero et al., 2014). For Cherry classification, Convolutional Numeral Network (CNN) with hybrid pooling method has been proposed to determine the appearance feature in regular or irregular shape with 99.4% of accuracy (Momeny, Jahanbakhshi, Jafarnezhad, & Zhang, 2020). For banana grading, Neural Network Arbitration has been developed to reduce human error and time which can determine if the banana is healthy or defective with 97% accuracy (Olaniyi, Oyedotun, & Adnan, 2017). The size of banana has been determined by using computer vision to find five point at the edge then measure the length and arc height (Hu, Dong, Malakar, Liu, & Jaganathan, 2015). Blueberry maturity has been classified by using histogram orientated gradients (HOG) which are mature, intermediate and young. The images were acquired from outdoor. Support Vector Machine (SVM) used to detect fruit region. K-nearest Neighbor (KNN) and Template Matching with Weighted Euclidean Distance (TMWE) has been used to classify the maturity state with low computation cost and high accuracy of 86.0% for young, 94.2% for intermediate and 96.0% for mature (Tan, Lee, Gan, & Wang, 2018). Artificial neural networks (ANNs) and support vector machine (SVM) has been developed for determining mulberry ripeness level. The image has been segmented in RGB channel which B was the best channel to classify the fruits. Color, geometric and texture features has been extracted by using Correlation-based Feature Selection subset (CFS) and Consistency subsets (CONS). ANN and SVM have been used for classification. ANN with CFS gave the best result with more minor error which are 100%, 100% and 99.1% of accuracy (Azarmdel, Jahanbakhshi, & Muñoz, 2020).

2.2 Research Gap

Several works have been studied from the literature review chapter which image processing and machine learning have been improved for fruits and vegetable quality assessment. This research will mainly focus on the machine learning part. There are various methods proposed recently. For instance, Fuzzy C-mean (FCM), Support vector machine (SVM), Artificial neural network (ANN), K-mean clustering and Convolutional Neural Network (CNN). Most research has been focused on appearance features. For instance, size, defect, color and ripeness. The objectives of machine vision have been discussed that quality inspection is needed before setting a price but there is no research about purchasing opportunity. So, this research objective is to classify the saleability of the lime by its external features. The summary of the related work is shown in the table 2.1.

Table 2.1 Summary of related research.

Research Paper	Objective	Method
Phate et al., 2021	Weight of lime	ANFIS
Phate et al., 2020	Weight of lime	SVM
ASabzi et al., 2020	PH value of orange	ANN
Momeny et al., 2020	Cherry classification	CNN
Phate et al., 2019	Weight of lime	FCM, SC, GP, ANFIS
Azarmdel et al., 2020	Mulberry ripeness level	SVM, ANN
Nyalala et al., 2019	Mass and volume of cherry tomatoes	SVM, ANN, RBF
Ileri et al., 2019	Defect of tomatoes	SVM, RBF
Wan et al., 2018	Maturity of tomatoes	BPNN
Behera et al., 2018	Severity disease	K-mean, SVM
Tan et al., 2018	Blueberry maturity	SVM, KNN
Olaniyi et al., 2017	Banana classification	BPNN
Arakeri, 2016	Defective and ripeness of tomatoes	ANN
Hu et al., 2015	Size of banana	Computer vision
Cubero et al., 2014	Grading of citrus	Computer vision
This research	Classifying saleability of lime	CNN

2.3 Convolutional Neural Network (CNN)

The Artificial Neural Network (ANN) is a computer processing system that based on the biological nervous system. There are composed of many connected nodes as known as neurons. ANN has ability to learn from input to optimise the final output. The input usually in multidimensional data which will feed to the hidden layer and it will learn from previous layer to improve the output. Two main methods for training neural network are supervised and unsupervised learning. Supervised learning has the labeled input. Otherwise, unsupervised learning has no labels. The image pattern recognition is better to use supervised learning (O'Shea & Nash, 2015). For pattern recognition, CNN is one of the most famous in deep neural network. In machine learning issues, the CNN performs effectively in applications that particular deal with image data such as face detection, image or video recognition. The layers of CNN consist of convolutional layer, non-linearity layer, pooling layer and fully connected layer (Eremenko, 2018). The distinction between CNN and ANN is CNN can recognize patterns in images. It enables the model to encode a specific feature from the input images which more suitable for image data (Albawi, Mohammed, & Al-Zawi, 2017).

2.3.1 Structure of Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) consists of four layers which are convolutional layer, pooling layer, flattening layer and fully connected layer is shown in figure 2.1.

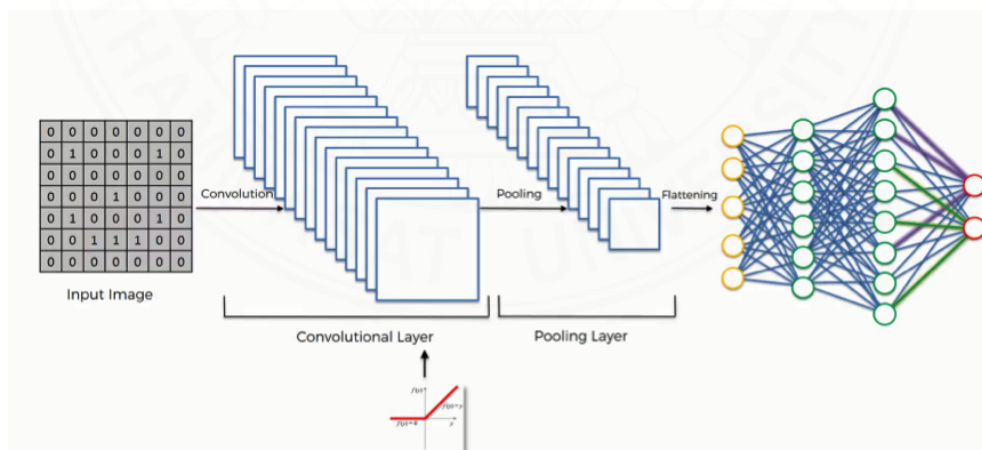


Figure 2.1 Layers of CNN model (Eremenko, 2018).

2.3.1.1 Convolutional Layer

Convolutional layer is used to reduce input image size and also perform to sharpen image, edge detection, or blur image. There is a feature detector or a filter which will apply in every pixels of input image and the result will be stored in feature map is shown in figure 2.2.

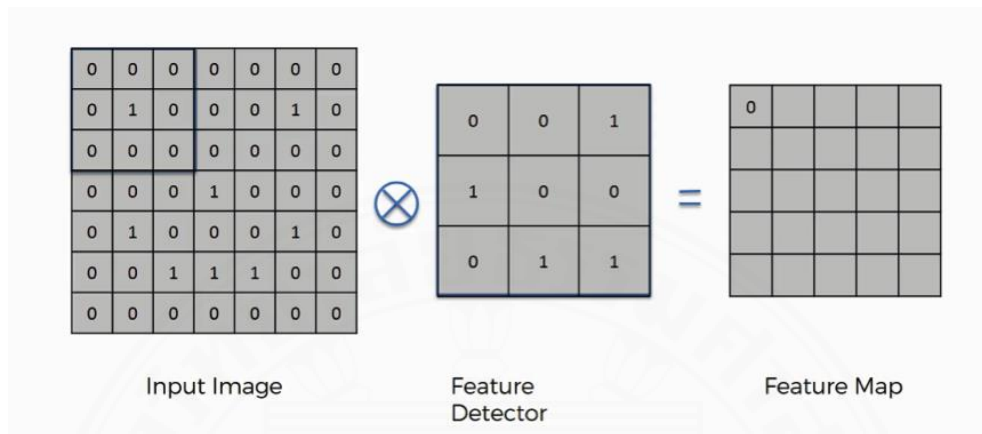


Figure 2.2 Convolutional layer (Eremenko, 2018).

2.3.1.2 Pooling Layer

In pooling layer, the filter will apply in every pixels of feature map image from convolutional process. In this step, the maximum value will be picked and store in pooled feature map. This step can help model to be robust with different manner of images. The process of this step is shown in figure 2.3.

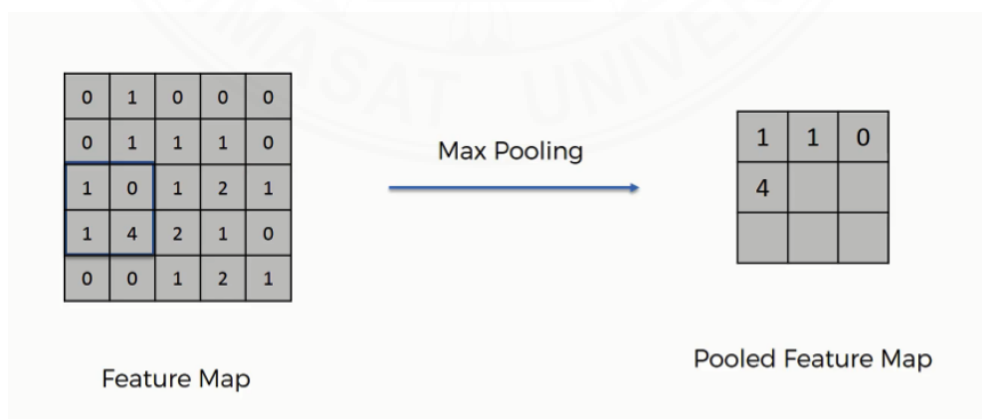


Figure 2.3 Pooling layer (Eremenko, 2018).

2.3.1.3 Flattening Layer

The output from pooling layer will be flattened in to one dimension to be able to perform as an input for Artificial Neural Network (ANN) model. The process of this step is shown in figure 2.4.

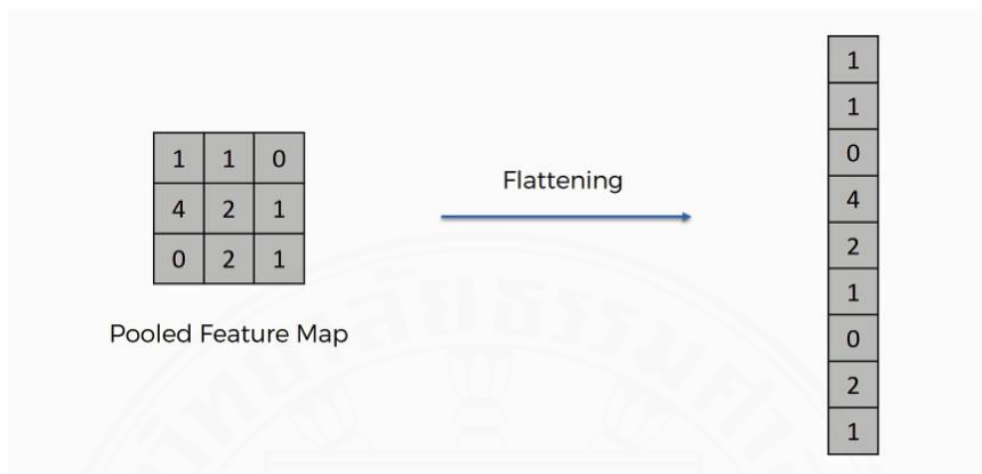


Figure 2.4 Flattening layer (Eremenko, 2018).

2.3.1.4 Fully Connected Layer

In the last step, the data will be fed into an ANN model which consists of input layer, fully connection layer or hidden layer and output layer. The output will show the value of most likely of input image and the model will selected the best value to be the label of that image. The process of this step is shown in figure 2.5.

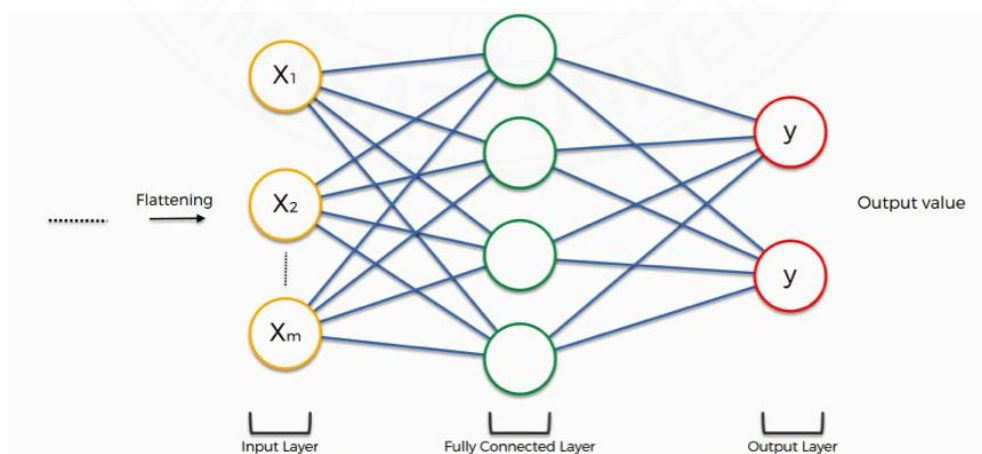


Figure 2.5 Fully connected layer (Eremenko, 2018).

2.3.2 Performance Evaluation

To evaluate the model performance, precision, recall, F1-score have been used in this section (Leung, 2022).

2.3.2.1 Precision

The precision is calculated by number of true positive divided by total positive. The formula for calculating precision is shown in equation 2.1.

$$Precision = \frac{TP}{TP + FP} \quad (2.1)$$

2.3.2.2 Recall

The recall is calculated by number of true positive divided by total of true positive and false negative. The formula for calculating recall is shown in equation 2.2.

$$Recall = \frac{TP}{TP + FN} \quad (2.2)$$

2.3.2.3 F1-score

The F1-score is calculated by precision and recall harmonic mean. The formula for calculating F1-score is shown in equation 2.3.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.3)$$

Where TP is True positive, TN is True negative, FP is False positive and FN is False negative.

CHAPTER 3 METHODOLOGY

The methodology section has divided into four steps which are data collection, data preparation, Convolutional Neural Network (CNN) model and hyperparameters tuning. The overall process is shown in figure 3.1.

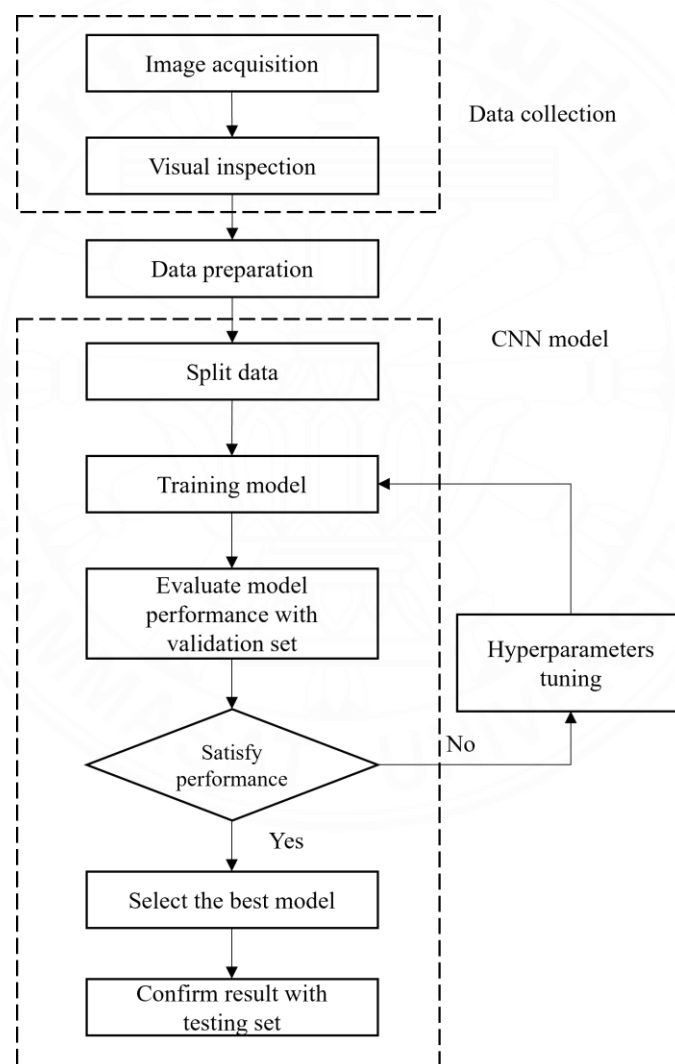


Figure 3.1 Methodology process.

3.1 Data Collection

Firstly, the data has to be collected before classification process. For data collection section, they are separated into 3 parts which are image acquisition, visual inspection, and data combination.

3.1.1 Image Acquisition

In image acquisition section, 530 various qualities of limes are collected from the farm. The limes that have been used in this research is called *Cirtus Aurantifolia* Swingle or *Citrus Aurantifolia* (Christm & Panze) Swing from Ratchaburi province. After that, each of lime was captured in four angles images. To setup the equipment there are a lightbox for controlling the light, an iPhone 11 Pro Max camera that has resolution of 3024×4032 pixels and a tripod for holding an iPhone to make the distance equally from camera to lime sample. The equipment setup and example of four angles of a lime image is shown in figure 3.2 and 3.3.



Figure 3.2 Equipment setup.

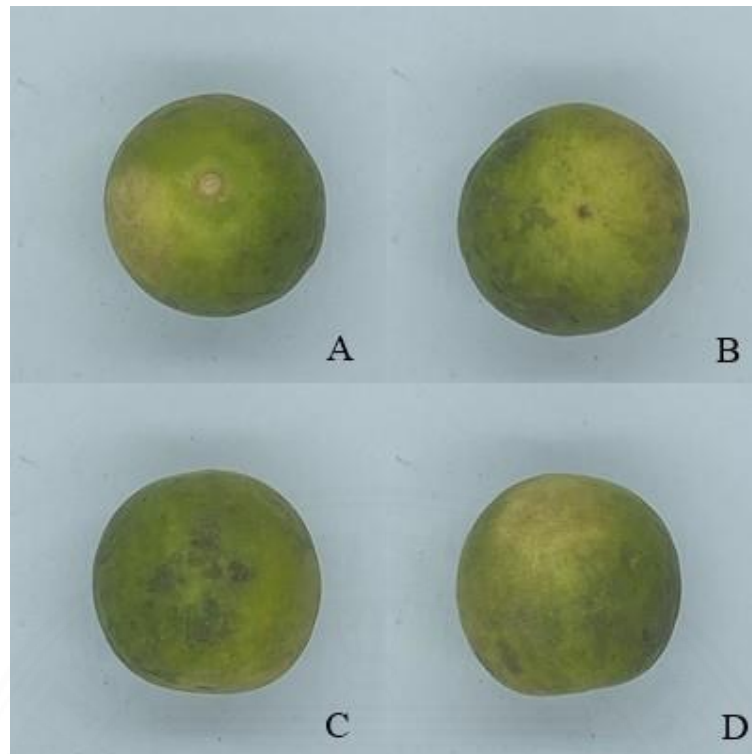


Figure 3.3 Four angles of lime image.

3.1.2 Visual Inspection

The survey was constructed and given to eight research participants in order to collect data for labeling the lime images. Each research participants were assigned to give the score range from 1 to 3 (1 is buy, 2 is maybe, 3 is not buy) from visual inspection at each angle of a lime images. The data collection table example for visual inspection is shown in table 3.1.

Table 3.1 Visual inspection table.

Lime number	Angle image	Saleability	Remark
1	A	1	Buy
1	B	2	Maybe
1	C	1	Buy
1	D	3	Not buy

3.2 Data Preparation

After acquired all data from the survey, mode is used to identify the label of lime images. The most frequently score that given from every researcher participants has been used to categorise the images into three different folders (buy, maybe, and not buy). The lime image size was reduced to 1000×1000 pixels to reduce the computation time and avoid reaching limitation of GPU memory.

3.3 Convolutional Neural Network (CNN) Model

In this section, CNN has been used to classify the lime saleability into 3 categories which are buy, maybe and not buy. Since the input data are images, CNN is one of the best model in the field of computer vision in image pattern recognition. The CNN was trained in Google Colaboratory. The code of this model is shown in appendix C.

3.3.1 Structure of Convolutional Neural Network (CNN) Model

The structure of CNN model consists of convolutional layer, pooling layer and fully connected layer. First, the input image was convoluted with the feature detector or filter size 3×3 . Then in pooling layer, the filter 2×2 was applied and stored the maximum value in the pooled feature map. After that the flattening process was applied to be an input of the fully connected layer which has dense layer and output layer (Sudha & Aji, 2021). In this layer, the images will be categorised in to three categories which are buy, maybe and not buy. The flow chart of CNN structure is shown in figure 3.4.

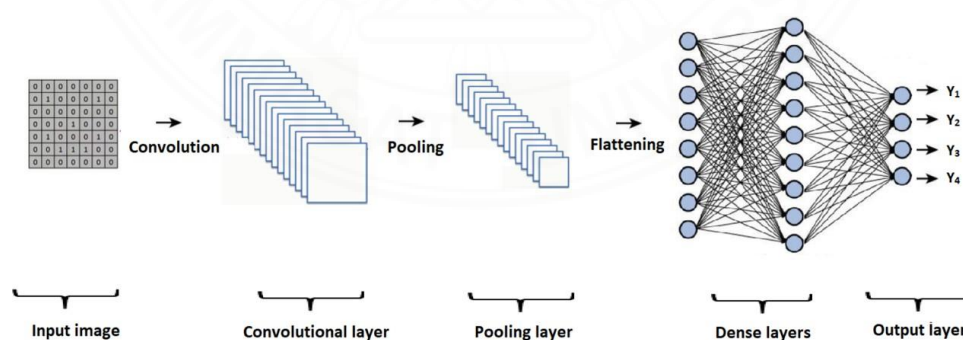


Figure 3.4 Structure of CNN model (Sudha & Aji, 2021).

3.3.2 Process of Convolutional Neural Network (CNN)

The data that fed into the CNN model consists of total 1128 lime images in three categories which are equally contain 376 images in buy, maybe and not buy. After that, the data was split into three portions of 80:10:10. The first 80 percent is for training set and the last 10 percent is for validation and testing set respectively. Training set is used to feed in the model for model training process, validation set is used to see how the model works after process of tuning hyperparameters and searching for the output that gives the most satisfy performance and testing set is used to experiment the best model that was selected to test model performance with the real-world data or the data that model never seen before. After data has split, training data is fed into CNN model for training process in this step the model will learn all images and classify the images in to three categories which are buy, maybe, and not buy. After the model is trained completely, the model performance will be evaluated with the data from validation set. The performance of each trained models will be compared with several experiments to find the best model for categorising lime's saleablilty. To obtain the optimal value, hyperparameters tuning process is needed for parameters searching. If the performance still not satisfy, hyperparameters tuning process will be applied until the best performance is acquired. Finally, the model that gave the best performance in validation set and had already confirmed the results with the testing set will be selected as the final model. The flowchart of model training process is shown in figure 3.5.

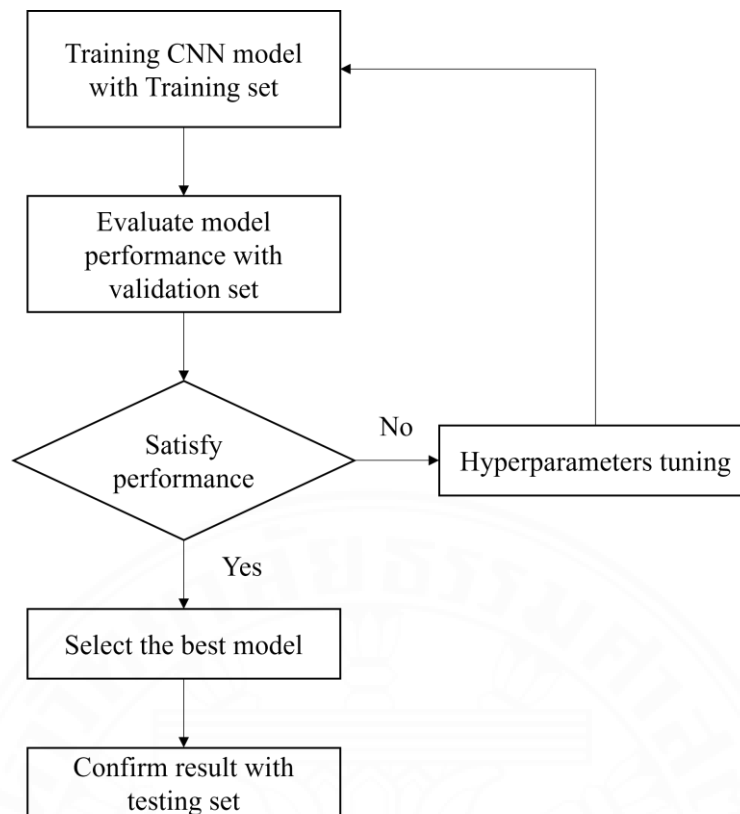


Figure 3.5 Training process of model.

3.3.3 Model Selection

The model performance evaluation methods that have been used in this research are recall, precision and F1-score. The performance that mostly focused was the F1-score because it is the combination of recall and precision which should be considered in term of model evaluation. It is calculated from the harmonic mean from both of them and provides more balance model performance for summarising the model. After that, accuracy, macro average and weighted average were calculated to combine the three performances from three categories which are buy, maybe and not buy. The method that mostly focus on was weighted average which consider each classes contribution or the support. F1-score will be calculated according to its size while macro average will treat all classes equally. So, in this research will be mainly focus on weighted average of F1-score for model evaluation process and the best model will be selected based on this criteria.

3.4 Hyperparameters Tuning

In hyperparameters tuning section, the experiment was constructed by using Design of Experiment (DOE) method to search for an optimal set of hyperparameters which give the best model performance. First experiment is constructed by using 2^5 full factorial design as initial experiment. The performances of each set of hyperparameters setting were collected in table of train recall, train precision, train F1-score, validate recall, validate precision, validate F1-score, test recall, test precision and test F1-score. After obtaining the performances from the first round, next experiment will be constructed based on the first experiment result which analysed by using Minitab.

3.4.1 Initial Full Factorial Design

To construct the first experiment for tuning the model, initial 2^5 full factorial design has been used in this step. There are five hyperparameters have been considered which are Batch size, Epoch, Learning rate, Decaying rate, and Momentum. Each hyperparameters was set low and high value as follow. Batch size is set to 64 and 128. Epoch is set to 250 and 500. Learning rate is set to 0.0125 and 0.0250. Decaying rate is set to 0.00125 and 0.00250. Momentum is set to 0.6 and 0.9. The low and high value setting is shown in table 3.2. The list of hyperparameters in the first experiment is shown in table 3.3.

Table 3.2 2^5 full factorial design low and high value setting.

hyperparameters	Low	High
Batch size	64	128
Epoch	250	500
Learning rate	0.0125	0.0250
Decaying rate	0.00125	0.00250
Momentum	0.6	0.9

Table 3.3 Hyperparameters setting table in initial full factorial design.

RunOrder	Batch size	Epoch	Learning rate	Decaying Rate	Momentum
1	64	250	0.0125	0.00125	0.6
2	128	250	0.0125	0.00125	0.6
3	64	500	0.0125	0.00125	0.6
4	128	500	0.0125	0.00125	0.6
5	64	250	0.025	0.00125	0.6
6	128	250	0.025	0.00125	0.6
7	64	500	0.025	0.00125	0.6
8	128	500	0.025	0.00125	0.6
9	64	250	0.0125	0.0025	0.6
10	128	250	0.0125	0.0025	0.6
11	64	500	0.0125	0.0025	0.6
12	128	500	0.0125	0.0025	0.6
13	64	250	0.025	0.0025	0.6
14	128	250	0.025	0.0025	0.6
15	64	500	0.025	0.0025	0.6
16	128	500	0.025	0.0025	0.6
17	64	250	0.0125	0.00125	0.9
18	128	250	0.0125	0.00125	0.9
19	64	500	0.0125	0.00125	0.9
20	128	500	0.0125	0.00125	0.9
21	64	250	0.025	0.00125	0.9
22	128	250	0.025	0.00125	0.9
23	64	500	0.025	0.00125	0.9
24	128	500	0.025	0.00125	0.9
25	64	250	0.0125	0.0025	0.9
26	128	250	0.0125	0.0025	0.9
27	64	500	0.0125	0.0025	0.9
28	128	500	0.0125	0.0025	0.9
29	64	250	0.025	0.0025	0.9
30	128	250	0.025	0.0025	0.9
31	64	500	0.025	0.0025	0.9
32	128	500	0.025	0.0025	0.9
33	96	375	0.01875	0.001875	0.75
34	96	375	0.01875	0.001875	0.75
35	96	375	0.01875	0.001875	0.75
36	96	375	0.01875	0.001875	0.75
37	96	375	0.01875	0.001875	0.75

3.4.2 Reduced Full Factorial Design

After all performances from first experiment are collected, second experiment will be constructed based on 2^5 full factorial design that eliminated unimportant factors by using Minitab. After the performances from second experiment are collected, the first and second experiment will be compared. If the second experiment gives better performance than first experiment. The third experiment will be constructed. This process will be repeated until the best performances are acquired. The hyperparameters tuning process is shown in figure 3.6.

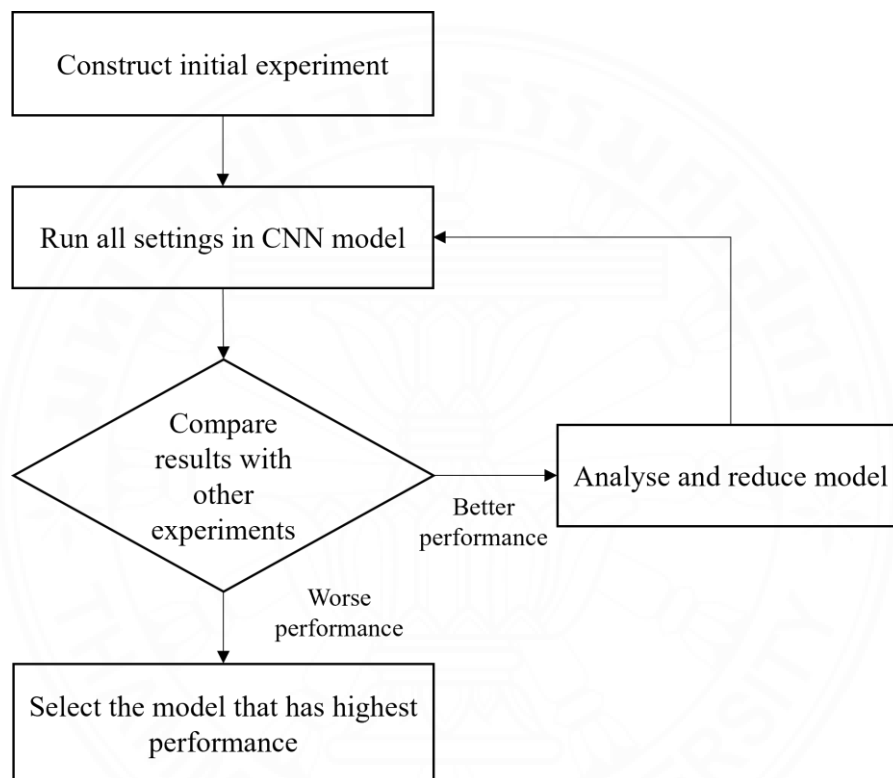


Figure 3.6 Hyperparameters tuning process.

CHAPTER 4

RESULT AND DISCUSSION

Nowadays, machine learning has been used to facilitate human in many applications. In this research, CNN has been developed to classify saleability of limes. The expected result of this research is the model can classify lime images into 3 categories which are buy, maybe and not buy that perform well with testing data set of lime images and give satisfied performance.

4.1 Result of Hyperparameters Tuning Process

To achieve the best performance, two hyperparameters tuning experiments were constructed. To evaluate the model performance, weighted average of F1-score has been used because it is one of the robust methods to evaluate the model performance of classification approach.

4.1.1 Result of Initial Full Factorial Design

After running the first experiment which constructed based on 2^5 full factorial design, all performances were collected. The performances from first experiment are shown in table B.1, B.2 and B.3 in appendix B. Hyperparameters set that gave the highest weighted average of F1-score was selected as the best performance in the first experiment. The summary of the best performance in first experiment is shown in table 4.1, 4.2 and 4.3.

Table 4.1 Summary result for training set in initial full factorial design.

Training set	Precision	Recall	F1-score	Support
Buy	67%	85%	75%	298
Maybe	62%	62%	62%	302
Not buy	100%	73%	85%	302
Accuracy			73%	902
Macro average	76%	73%	74%	902
Weighted average	76%	73%	74%	902

Table 4.2 Summary result for validation set in initial full factorial design.

Validation set	Precision	Recall	F1-score	Support
Buy	77%	77%	77%	44
Maybe	65%	76%	70%	37
Not buy	96%	78%	86%	32
Accuracy			77%	113
Macro average	80%	77%	78%	113
Weighted average	79%	77%	77%	113

Table 4.3 Summary result for testing set in initial full factorial design.

Testing set	Precision	Recall	F1-score	Support
Buy	72%	91%	81%	34
Maybe	64%	68%	66%	37
Not buy	100%	74%	85%	42
Accuracy			77%	113
Macro average	79%	78%	77%	113
Weighted average	80%	77%	77%	113

4.1.2 Result After Reduced Full Factorial Design

After all performances from the first experiment were collected, Minitab was used to analyse the performance in the first experiment to eliminate unimportant factors and second experiment was constructed based on 2^5 full factorial design. The hyperparameters table of second experiment is shown in table A.2 in appendix A. After run second experiment, the hyperparameters set that gave the highest weighted average of F1-score was selected as the best performance in second experiment. The summary of the best performance in second experiment is shown in table 4.4, 4.5 and 4.6. All performances from second experiment are shown in table B.4, B.5 and B.6 in appendix B.

Table 4.4 Summary result for training set after reduced full factorial design.

Training set	Precision	Recall	F1-score	Support
Buy	73%	74%	74%	298
Maybe	65%	66%	65%	302
Not buy	90%	88%	89%	302
Accuracy			76%	902
Macro average	76%	76%	76%	902
Weighted average	76%	76%	76%	902

Table 4.5 Summary result for validation set after reduced full factorial design.

Validation set	Precision	Recall	F1-score	Support
Buy	79%	68%	73%	44
Maybe	68%	68%	63%	37
Not buy	82%	84%	83%	32
Accuracy			73%	113
Macro average	73%	73%	73%	113
Weighted average	73%	73%	73%	113

Table 4.6 Summary result for testing set after reduced full factorial design.

Testing set	Precision	Recall	F1-score	Support
Buy	73%	79%	76%	34
Maybe	62%	57%	59%	37
Not buy	86%	86%	86%	42
Accuracy			74%	113
Macro average	73%	74%	74%	113
Weighted average	74%	74%	74%	113

4.1.3 Result Comparison

Comparing the weighted average of F1-score in validation set as shown in graph figure 4.1 and 4.2, first experiment gave better performance than second experiment. The performance in second experiment did not change much in the beginning and decreasing rapidly after run 16. So, the run order from first experiment that gave the highest weighted average of F1-score in validation set was selected to be the best performance of this research which indicated in run order 31. In training section, the F1-score is 75%, 62%, 85%. In validation section, the F1-score is 77%, 70%, 86%. In testing section, the F1-score is 81%, 66%, 85% for buy, maybe and not buy respectively. For weighted average of F1-score, training gives performance of 74% and 77% for validation and testing. To get these results, the hyperparameters were set as batch size is equal to 64, epoch is 500, learning rate is 0.025, decaying rate is 0.0025, and momentum is 0.9. The summary of best performance setting and result are shown in table 4.7 and 4.8.



Figure 4.1 Weighted average of F1-score in first experiment.

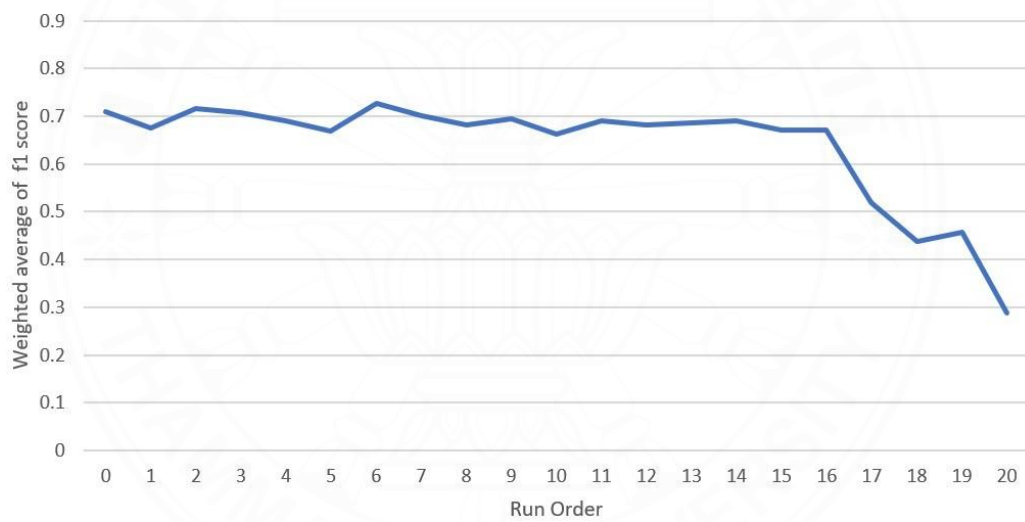


Figure 4.2 Weighted average of F1-score in second experiment.

Table 4.7 Best hyperparameters setting.

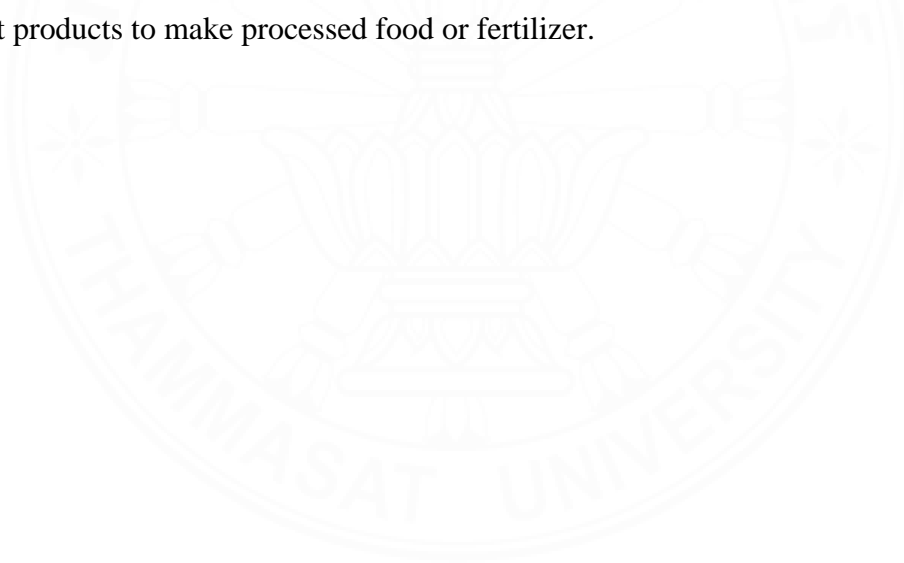
RunOrder	31
Batch Size	64
Epoch	500
Learning Rate	0.025
Decaying Rate	0.0025
Momentum	0.9

Table 4.8 Summary of the best F1-score result.

Category	F1 score		
	Train	Validation	Test
Buy	75 %	77 %	81 %
Maybe	62 %	70 %	66 %
Not buy	85 %	86 %	85 %
Weighted average	74 %	77 %	77 %

4.2 Discussion

From the summary result of the best performance that shown in table 4.8, the highest F1-score is not buy category which gave 85% on testing data set. It gave outstanding performance than buy and maybe categories. Indicated that this model is good for sorting out the unwanted limes. Lime is one of Thai economic crops that has high demand all the year. According to statistical research from Trade Policy and Strategy Office (TPSO) of Thailand, limes were produced 483,930 tons in 2020 and exported 1,413 tons or 24.55 millions Baht in 2021 (TPSO, 2021). Sorting the not buy or unwanted limes out from saleable limes is very important in quality inspection process before selling to customers since customers satisfaction is one of the main goals of many companies which they want to avoid selling bad quality of limes to the market. Selling bad quality of lime to customer may cause losing their customer bases. In this research, although there is a chance that the model will classify limes into wrong category which the 15% of saleable limes could be sorted out. Tradeoff between customer unsatisfied cost and losing some saleable limes, it is worthy to maintain customer satisfaction. For the unwanted lime, it can be sold in the lower price as defect products to make processed food or fertilizer.



CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The CNN model can classify lime images into three categories (buy, maybe, not buy). The model hyperparameters have been tuned to achieve the optimal performance for classifying lime images. Two experiments were constructed based on 2^5 full factorial method. The first experiment gave better performance than second experiment. The best performance gave F1-score in testing set of 81%, 66% and 85% in buy, maybe and not buy categories while the weighted average of F1-score is 77%. Since the not buy category gave higher performance than other two categories, indicated that the model is better in sorting unwanted lime out. This research demonstrated the effectiveness of CNN for agriculturists to classify saleability instead of using human or large machine which has high enough performance to classify the lime images into correct categories. The limitation of this research is that it has been experimented only with Citrus Aurantifolia Swingle lime samples and the equipment setting is used for experiment only which cannot perform in the real situation.

5.2 Future Work

For future work, this model can be adapted with real time camera. So, it can be used for lime classifying machine by connecting the camera that help human to detect the limes in conveyor. Moreover, it can be used in the mobile phone application to help customer to select the lime. Lastly, the performance could be improved by collecting more lime samples for training the CNN model.

REFERENCES

- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *international conference on engineering and technology (ICET)*, 1–6.
- Arakeri, M. P. (2016). Computer vision based fruit grading system for quality evaluation of tomato in agriculture industry. *Procedia Computer Science*, 79, 426–433.
- ASabzi, S., Javadikia, H., & Arribas, J. I. (2020). A three-variety automatic and non-intrusive computer vision system for the estimation of orange fruit ph value. *Measurement*, 152, 107298.
- Azarmdel, H., Jahanbakhshi, S., Aa nd Mohtasebi, & Muñoz, A. R. (2020). Evaluation of image processing technique as an expert system in mulberry fruit grading based on ripeness level using artificial neural networks (anns) and support vector machine (svm). *Postharvest Biology and Technology*, 166, 111201.
- Behera, S. K., Rath, A. K., Mahapatra, A., & Sethy, P. K. (2020). Identification, classification and grading of fruits using machine learning and computer intelligence: A review. *Journal of Ambient Intelligence and Humanized Computing*, 1–11.
- Behera, S., Jena, L., Rath, A., & Sethy, P. (2018). Disease classification and grading of orange using machine learning and fuzzy logic. *International Conference on Communication and Signal Processing (ICCSP)*, 0678–0682.
- Bhargava, A., & Bansal, A. (2021). Fruits and vegetables quality evaluation using computer vision: A review. *Journal of King Saud University-Computer and Information Sciences*, 33(3), 243–257.
- Booranawong, T., & Booranawong, A. (2017). Simple and double exponential smoothing methods with designed input data for forecasting a seasonal time series: In an application for lime prices in thailand. *Suranaree Journal of Science and Technology*, 24(3).
- Booranawong, T., & Booranawong, A. (2018). Double exponential smoothing and holt-winters methods with optimal initial values and weighting factors for forecasting lime, thai chili and lemongrass prices in thailand. *Engineering and Applied Science Research*, 45(1), 32–38.
- Cubero, S., Aleixos, N., Albert, F., Torregrosa, A., Ortiz, C., Garc'ia-Navarrete, O., & Blasco, J. (2014). Optimised computer vision system for automatic pre-grading of citrus fruit in the field using a mobile platform. *Precision Agriculture*, 15(1), 80–94.

- Eremenko, K. (2018). The ultimate guide to convolutional neural networks (cnn). Retrieved from <https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn>
- Hu, M. H., Dong, Q. L., Malakar, P. K., Liu, B. L., & Jaganathan, G. K. (2015). Determining banana size based on computer vision. *International journal of food properties*, *18*(3), 508–520.
- Ileri, D., Belal, E., Okinda, C., Makange, N., & Ji, C. (2019). A computer vision system for defect discrimination and grading in tomatoes using machine learning and image processing. *Artificial Intelligence in Agriculture*, *2*, 28–37.
- Kaewsuksaeng, S., Tatmala, N., Srilaong, V., & Pongprasert, N. (2015). Postharvest heat treatment delays chlorophyll degradation and maintains quality in thai lime (citrus aurantifolia swingle cv. paan) fruit. *Postharvest Biology and Technology*, *100*, 1–7.
- Leung, K. (2022). Micro, macro and weighted averages of F1 score, clearly explained. Retrieved from <https://towardsdatascience.com/micro-macro-weighted-averages-of-F1-score-clearly-explained-b603420b292f>
- Momeny, M., Jahanbakhshi, A., Jafarnejhad, K., & Zhang, Y. D. (2020). Accurate classification of cherry fruit using deep cnn based on hybrid pooling approach. *Postharvest Biology and Technology*, *166*, 111204.
- Narang, N., & Jiraungkoorskul, W. (2016). Anticancer activity of key lime, citrus aurantifolia. *Pharmacognosy reviews*, *10*(20), 118.
- Nyalala, I., Okinda, C., Nyalala, L., Makange, N., Chao, Q., Chao, L., & Chen, K. (2019). Tomato volume and mass estimation using computer vision and machine learning algorithms: Cherry tomato model. *Journal of Food Engineering*, *263*, 288–298.
- O’Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv, 1511*, 08458.
- Olaniyi, E. O., Oyedotun, O. K., & Adnan, K. (2017). Intelligent grading system for banana fruit using neural network arbitration. *Journal of Food Process Engineering*, *40*(1), e13160.
- Pathmanaban, P., Gnanavel, B. K., & Anandan, S. S. (2019). Recent application of imaging techniques for fruit quality assessment. *Trends in Food Science and Technology*, *94*, 32–42.
- Phate, V., Malmathanraj, R., & Palanisamy, P. (2019). Clustered anfis weighing models for sweet lime (citrus limetta) using computer vision system. *Journal of Food Process Engineering*, *42*(6), e13160.

- Phate, V., Malmathanraj, R., & Palanisamy, P. (2020). An indirect method to estimate sweet lime weight through machine learning algorithm. *Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, 194–198.
- Phate, V., Malmathanraj, R., & Palanisamy, P. (2021). Classification and indirect weighing of sweet lime fruit through machine learning and meta-heuristic approach. *International Journal of Fruit Science*, 21(1), 528–545.
- Saldaña, E., Siche, R., Luján, M., & Quevedo, R. (2013). Brazilian journal of food technology. *Pharmacognosy reviews*, 16, 254–272.
- Sudha, S. K., & Aji, S. (2021). An analysis on deep learning approaches: Addressing the challenges in remote sensing image retrieval. *International Journal of Remote Sensing*, 42(24), 9405–9441.
- Tan, K., Lee, W. S., Gan, H., & Wang, S. (2018). Recognising blueberry fruit of different maturity using histogram oriented gradients and colour features in outdoor scenes. *Biosystems engineering*, 176, 59–72.
- TPSO. (2021). Analysis of thai economic and trade situations. Retrieved from http://www.tpso.moc.go.th/sites/default/files/wiekhraahsthaankaarnesrsthkickchkaarkhaa_rwmeduue.pdf
- Wan, P., Toudeshki, A., Tan, H., & Ehsani, R. (2018). A methodology for fresh tomato maturity detection using computer vision. *Computers and electronics in agriculture*, 146, 43–50.



APPENDICES

APPENDIX A

TABLES OF HYPERPARAMETERS IN FULL FACTORIAL METHOD

Table A.1 First experiment for hyperparameter setting.

RunOrder	Batch size	Epoch	Learning rate	Decaying Rate	Momentum
1	64	250	0.0125	0.00125	0.6
2	128	250	0.0125	0.00125	0.6
3	64	500	0.0125	0.00125	0.6
4	128	500	0.0125	0.00125	0.6
5	64	250	0.025	0.00125	0.6
6	128	250	0.025	0.00125	0.6
7	64	500	0.025	0.00125	0.6
8	128	500	0.025	0.00125	0.6
9	64	250	0.0125	0.0025	0.6
10	128	250	0.0125	0.0025	0.6
11	64	500	0.0125	0.0025	0.6
12	128	500	0.0125	0.0025	0.6
13	64	250	0.025	0.0025	0.6
14	128	250	0.025	0.0025	0.6
15	64	500	0.025	0.0025	0.6
16	128	500	0.025	0.0025	0.6
17	64	250	0.0125	0.00125	0.9
18	128	250	0.0125	0.00125	0.9
19	64	500	0.0125	0.00125	0.9
20	128	500	0.0125	0.00125	0.9
21	64	250	0.025	0.00125	0.9
22	128	250	0.025	0.00125	0.9
23	64	500	0.025	0.00125	0.9
24	128	500	0.025	0.00125	0.9
25	64	250	0.0125	0.0025	0.9
26	128	250	0.0125	0.0025	0.9
27	64	500	0.0125	0.0025	0.9
28	128	500	0.0125	0.0025	0.9
29	64	250	0.025	0.0025	0.9
30	128	250	0.025	0.0025	0.9
31	64	500	0.025	0.0025	0.9
32	128	500	0.025	0.0025	0.9
33	96	375	0.01875	0.001875	0.75
34	96	375	0.01875	0.001875	0.75
35	96	375	0.01875	0.001875	0.75
36	96	375	0.01875	0.001875	0.75
37	96	375	0.01875	0.001875	0.75

Table A.2 Second experiment for hyperparameter setting.

RunOrder	Batch size	Epoch	Learning rate	Decaying Rate	Momentum
0	96	375	0.01875	0.001875	0.75
1	96	421	0.017827	0.001992	0.725
2	96	467	0.016904	0.002109	0.7
3	96	513	0.015982	0.002227	0.675
4	96	559	0.015059	0.002344	0.65
5	96	605	0.014136	0.002461	0.625
6	96	651	0.013213	0.002578	0.6
7	96	698	0.012291	0.002695	0.575
8	96	744	0.011368	0.002813	0.55
9	96	790	0.010445	0.00293	0.525
10	96	836	0.009522	0.003047	0.5
11	96	882	0.008599	0.003164	0.475
12	96	928	0.007677	0.003281	0.45
13	96	974	0.006754	0.003398	0.425
14	96	1020	0.005831	0.003516	0.4
15	96	1066	0.004908	0.003633	0.375
16	96	1112	0.003986	0.00375	0.35
17	96	1158	0.003063	0.003867	0.325
18	96	1204	0.00214	0.003984	0.3
19	96	1251	0.001217	0.004102	0.275
20	96	1297	0.000294	0.004219	0.25



APPENDIX B

RESULTS OF HYPERPARAMETERS TUNING PROCESS

Table B.1 Training result from first experiment.

RunOrder	Train recall	Train precision	Train F1 score
1	0.701	0.665	0.658
2	0.680	0.645	0.638
3	0.749	0.725	0.726
4	0.675	0.633	0.620
5	0.749	0.743	0.745
6	0.618	0.590	0.556
7	0.657	0.608	0.610
8	0.656	0.614	0.608
9	0.712	0.690	0.681
10	0.664	0.620	0.604
11	0.706	0.683	0.676
12	0.688	0.661	0.658
13	0.735	0.735	0.732
14	0.659	0.626	0.607
15	0.642	0.605	0.597
16	0.682	0.647	0.652
17	0.624	0.579	0.535
18	0.664	0.634	0.642
19	0.826	0.814	0.817
20	0.737	0.714	0.714
21	0.328	0.490	0.391
22	0.477	0.486	0.410
23	0.654	0.618	0.614
24	0.607	0.609	0.592
25	0.723	0.694	0.687
26	0.661	0.631	0.617
27	0.762	0.745	0.750
28	0.754	0.743	0.744
29	0.637	0.600	0.566
30	0.727	0.711	0.711
31	0.762	0.733	0.737
32	0.658	0.612	0.605
33	0.798	0.789	0.791
34	0.764	0.751	0.754
35	0.728	0.704	0.707
36	0.735	0.721	0.723
37	0.767	0.757	0.760

Table B.2 Validation result from first experiment.

RunOrder	Validate recall	Validate precision	Validate F1 score
1	0.701	0.690	0.685
2	0.724	0.717	0.710
3	0.705	0.699	0.700
4	0.646	0.637	0.616
5	0.706	0.690	0.693
6	0.624	0.628	0.608
7	0.772	0.761	0.763
8	0.679	0.673	0.664
9	0.695	0.690	0.689
10	0.730	0.708	0.699
11	0.704	0.699	0.697
12	0.698	0.699	0.694
13	0.650	0.646	0.647
14	0.635	0.637	0.615
15	0.752	0.743	0.739
16	0.772	0.761	0.763
17	0.637	0.619	0.588
18	0.722	0.717	0.719
19	0.751	0.726	0.730
20	0.739	0.735	0.734
21	0.304	0.434	0.349
22	0.441	0.487	0.418
23	0.693	0.690	0.682
24	0.699	0.708	0.697
25	0.681	0.681	0.671
26	0.655	0.655	0.642
27	0.698	0.681	0.686
28	0.688	0.690	0.688
29	0.646	0.637	0.597
30	0.707	0.708	0.707
31	0.786	0.770	0.774
32	0.732	0.717	0.710
33	0.705	0.699	0.701
34	0.691	0.681	0.677
35	0.689	0.681	0.683
36	0.705	0.708	0.705
37	0.699	0.681	0.684

Table B.3 Testing result from first experiment.

RunOrder	Test recall	Test precision	Test F1 score
1	0.734	0.708	0.695
2	0.732	0.717	0.708
3	0.728	0.708	0.706
4	0.656	0.646	0.620
5	0.743	0.743	0.740
6	0.648	0.628	0.584
7	0.706	0.681	0.677
8	0.677	0.664	0.648
9	0.726	0.717	0.706
10	0.667	0.655	0.637
11	0.702	0.699	0.674
12	0.723	0.717	0.708
13	0.722	0.702	0.708
14	0.669	0.664	0.637
15	0.704	0.681	0.679
16	0.730	0.708	0.708
17	0.615	0.602	0.560
18	0.692	0.681	0.680
19	0.815	0.805	0.809
20	0.730	0.717	0.715
21	0.326	0.487	0.390
22	0.617	0.558	0.478
23	0.743	0.726	0.717
24	0.638	0.655	0.621
25	0.721	0.708	0.694
26	0.662	0.655	0.634
27	0.750	0.743	0.744
28	0.735	0.735	0.729
29	0.666	0.646	0.616
30	0.717	0.708	0.704
31	0.798	0.770	0.773
32	0.688	0.664	0.653
33	0.756	0.743	0.744
34	0.745	0.717	0.714
35	0.746	0.735	0.732
36	0.733	0.726	0.722
37	0.780	0.770	0.774

Table B.4 Training result from second experiment.

RunOrder	Train recall	Train precision	Train F1 score
0	0.748	0.746	0.746
1	0.721	0.700	0.701
2	0.788	0.783	0.784
3	0.790	0.778	0.779
4	0.758	0.737	0.738
5	0.752	0.738	0.738
6	0.760	0.758	0.759
7	0.755	0.739	0.743
8	0.775	0.769	0.771
9	0.774	0.762	0.766
10	0.681	0.641	0.627
11	0.727	0.705	0.706
12	0.752	0.734	0.738
13	0.704	0.684	0.684
14	0.696	0.663	0.662
15	0.697	0.675	0.673
16	0.687	0.663	0.665
17	0.640	0.616	0.620
18	0.585	0.562	0.565
19	0.534	0.542	0.534
20	0.266	0.410	0.320

Table B.5 Validation result from second experiment.

RunOrder	Validate recall	Validate precision	Validate F1 score
0	0.713	0.708	0.709
1	0.681	0.673	0.675
2	0.715	0.717	0.716
3	0.713	0.708	0.707
4	0.700	0.690	0.691
5	0.673	0.673	0.669
6	0.734	0.726	0.727
7	0.705	0.699	0.701
8	0.682	0.681	0.681
9	0.701	0.690	0.694
10	0.678	0.673	0.662
11	0.695	0.690	0.690
12	0.682	0.681	0.681
13	0.690	0.690	0.687
14	0.693	0.690	0.690
15	0.671	0.673	0.671
16	0.670	0.673	0.671
17	0.514	0.522	0.518
18	0.442	0.434	0.437
19	0.453	0.469	0.457
20	0.236	0.389	0.289

Table B.6 Testing result from second experiment.

RunOrder	Test recall	Test precision	Test F1 score
0	0.734	0.735	0.732
1	0.752	0.743	0.739
2	0.728	0.726	0.721
3	0.727	0.717	0.714
4	0.708	0.681	0.679
5	0.721	0.717	0.713
6	0.740	0.743	0.741
7	0.735	0.726	0.726
8	0.752	0.752	0.750
9	0.740	0.735	0.735
10	0.690	0.681	0.660
11	0.687	0.681	0.670
12	0.714	0.708	0.708
13	0.727	0.717	0.711
14	0.727	0.717	0.705
15	0.732	0.726	0.713
16	0.724	0.717	0.714
17	0.663	0.646	0.647
18	0.519	0.504	0.505
19	0.500	0.522	0.507
20	0.316	0.469	0.376

APPENDIX C

PYTHON CODE OF CNN MODEL

```
1
2
3
4 # CNN model for predicting saleability
5
6 # 1. Model preparation
7
8
9 from keras.preprocessing.image import img_to_array
10
11 class ImageToArrayPreprocessor:
12     def __init__(self, dataFormat=None):
13         # store the image data format
14         self.dataFormat = dataFormat
15
16     def preprocess(self, image):
17         # apply the Keras utility function that correctly rearranges
18         # the dimensions of the image
19         return img_to_array(image, data_format=self.dataFormat)
20
21 import numpy as np
22 import cv2
23 import os
24
25 class SimpleDatasetLoader:
26     def __init__(self, preprocessors=None):
27         # store the image preprocessor
28         self.preprocessors = preprocessors
29
30         # if the preprocessors are None, initialize them as an
31         # empty list
32         if self.preprocessors is None:
33             self.preprocessors = []
34
35     def load(self, imagePaths, verbose=-1):
36         # initialize the list of features and labels
37         data = []
```



```

38     labels = []
39
40 # loop over the input images
41     for (i, imagePath) in enumerate(imagePaths):
42 # load the image and extract the class label assuming
43 # that our path has the following format:
44 # /path/to/dataset/{class}/{image}.jpg
45         image = cv2.imread(imagePath)
46         label = imagePath.split(os.path.sep)[-2]
47 # check to see if our preprocessors are not None
48         if self.preprocessors is not None:
49
50             for p in self.preprocessors:
51                 image = p.preprocess(image)
52
53 # treat our processed image as a "feature vector"
54 # by updating the data list followed by the labels
55
56         data.append(image)
57         labels.append(label)
58
59 # show an update every verbose images
60         if verbose > 0 and i > 0 and (i + 1) % verbose == 0:
61             print("[INFO] processed {}/{}".format(i + 1, len(imagePaths)))
62
63 # return a tuple of the data and labels
64         return (np.array(data), np.array(labels))
65
66 import imutils
67 import cv2
68
69 class AspectAwarePreprocessor:
70     def __init__(self, width, height, inter=cv2.INTER_AREA):
71 # store the target image width, height, and interpolation
72 # method used when resizing
73         self.width = width
74         self.height = height
75         self.inter = inter
76
77     def preprocess(self, image):
78 # grab the dimensions of the image and then initialize

```

```

79 # the deltas to use when cropping
80     (h, w) = image.shape[:2]
81     dW = 0
82     dH = 0
83
84 # if the width is smaller than the height, then resize
85 # along the width (i.e., the smaller dimension) and then
86 # update the deltas to crop the height to the desired
87 # dimension
88     if w < h:
89         image = imutils.resize(image, width=self.width,
90                                inter=self.inter)
91         dH = int((image.shape[0] - self.height) / 2.0)
92
93 # otherwise, the height is smaller than the width so
94 # resize along the height and then update the deltas
95 # to crop along the width
96     else:
97         image = imutils.resize(image, height=self.height,
98                                inter=self.inter)
99         dW = int((image.shape[1] - self.width) / 2.0)
100
101 # now that our images have been resized, we need to
102 # re-grab the width and height, followed by performing
103 # the crop
104     (h, w) = image.shape[:2]
105     image = image[dH:h - dH, dW:w - dW]
106
107 # finally, resize the image to the provided spatial
108 # dimensions to ensure our output image is always a fixed
109 # size
110     return cv2.resize(image, (self.width, self.height),
111                       interpolation=self.inter)
112
113 from sklearn.metrics import classification_report
114 from keras.models import Sequential
115 from keras.layers.convolutional import Conv2D
116 from keras.layers.convolutional import MaxPooling2D
117 from keras.layers.core import Activation
118 from keras.layers.core import Dense
119 from keras.layers.core import Flatten

```

```
120 import matplotlib.pyplot as plt
121 from tensorflow import keras
122 from keras.applications import imagenet_utils
123 from tensorflow.keras.preprocessing import image_dataset_from_directory
124 from keras.preprocessing.image import ImageDataGenerator
125 from keras.layers import Rescaling
126 import numpy as np
127 import argparse
128 import cv2
129 from imutils import paths
130 from sklearn.model_selection import train_test_split
131 from sklearn.preprocessing import LabelBinarizer
132 from keras.callbacks import ModelCheckpoint
133 from keras.models import load_model
134
135 """## 2. Parameters setting"""
136
137 from google.colab import drive
138 drive.mount('/content/drive')
139
140 #Parameters setting section
141 -----
142
143 #Model Train Parameters (the values are obtained from the DOE)
144 BATCH_SIZE = 96
145 ep = 175
146 Learning_Rate = 0.031893
147 Decay_Rate = 0.000781
148 Momentum_Rate = 0.99
149
150 # create a weight file name based on settings of an experiments
151 weight_f = 'CNN_saleability_expt_BATCH_%d_EP_%d_LR_%f_DECAY_%f_MOMENT_%f.h5'%(
152     BATCH_SIZE,
153     ep,
154     Learning_Rate,
155     Decay_Rate,
```

```
    Momentum_Rate)
155 print(weight_f) # for debugging purpose
156
157 # create a report file name based on settings of an experiments
158 train_report_f = 'train_CNN_saleability_expt_BATCH_%d_EP_%d_LR_%f_DECAY_%f_MOMENT_%
    f.csv'%(BATCH_SIZE,
159
    ep,
160
    Learning_Rate,
161
    Decay_Rate,
162
    Momentum_Rate)
163 print(train_report_f) # for debugging purpose
164
165 test_report_f = 'test_CNN_saleability_expt_BATCH_%d_EP_%d_LR_%f_DECAY_%f_MOMENT_%f.
    csv'%(BATCH_SIZE,
166
    ep,
167
    Learning_Rate,
168
    Decay_Rate,
169
    Momentum_Rate)
170 print(test_report_f) # for debugging purpose
171
172 val_report_f = 'val_CNN_saleability_expt_BATCH_%d_EP_%d_LR_%f_DECAY_%f_MOMENT_%f.
    csv'%(BATCH_SIZE,
173
    ep,
174
    Learning_Rate,
175
    Decay_Rate,
176
    Momentum_Rate)
177 print(val_report_f) # for debugging purpose
178
179 #Image File Directories
```

```

180
181 # This is the location of the entire dataset
182 image_path_train = '/content/drive/MyDrive/CNN_saleability/v3'
183
184 #Weight Save Directory
185 Weight_Save_Path = '/content/drive/MyDrive/CNN_saleability/v3_weights/Round2' +
    weight_f
186
187 #Report Directory
188 train_report_path = '/content/drive/MyDrive/CNN_saleability/v3_reports/Round2' +
    train_report_f
189 test_report_path = '/content/drive/MyDrive/CNN_saleability/v3_reports/Round2' +
    test_report_f
190 val_report_path = '/content/drive/MyDrive/CNN_saleability/v3_reports/Round2' +
    val_report_f
191
192 """## 3. Model training"""
193
194 #Model Training Section
    -----
195
196 imagePaths = list(paths.list_images(image_path_train))
197
198 # initialize the image preprocessors
199 sp = AspectAwarePreprocessor(128,128)
200 iap = ImageToArrayPreprocessor()
201
202 # load the dataset from disk then scale the raw pixel intensities
203 # to the range [0, 1]
204 sdl = SimpleDatasetLoader(preprocessors=[sp, iap])
205 (data, labels) = sdl.load(imagePaths, verbose=500)
206 data = data.astype("float") / 255.0
207
208 # set aside 10% of train and test data for evaluation
209 X_train, testX, Y_train, testY = train_test_split(data, labels,
210     test_size=0.1, random_state = 32)
211
212 # Use the same function above for the validation set
213 trainX, valX, trainY, valY = train_test_split(X_train, Y_train,
214     test_size=1/9, random_state= 32) # 1/9 x 0.9 = 0.1

```

```
215
216 # convert the labels from integers to vectors
217 trainY = LabelBinarizer().fit_transform(trainY)
218 valY = LabelBinarizer().fit_transform(valY)
219 testY = LabelBinarizer().fit_transform(testY)
220
221 # construct the image generator for data augmentation
222 aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
223     height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
224     horizontal_flip=True, fill_mode="nearest")
225
226
227 classes = 3
228 batch = BATCH_SIZE
229 chanDim = 1
230
231 # define model
232 model = Sequential()
233 model.add(Conv2D(64,(3,3),padding="same", input_shape=(128,128, 3)))
234 model.add(Activation('relu'))
235 model.add(Conv2D(64, (3, 3), padding="same"))
236 model.add(Activation("relu"))
237 model.add(MaxPooling2D(pool_size=(2, 2)))
238
239 model.add(Conv2D(128,(3,3),padding="same"))
240 model.add(Activation('relu'))
241 model.add(Conv2D(128, (3, 3), padding="same"))
242 model.add(Activation("relu"))
243 model.add(MaxPooling2D(pool_size=(2, 2)))
244
245 model.add(Conv2D(256,(3,3),padding="same"))
246 model.add(Activation('relu'))
247 model.add(Conv2D(256, (3, 3), padding="same"))
248 model.add(Activation("relu"))
249 #model.add(Conv2D(256,(3,3),padding="same"))
250 #model.add(Activation('relu'))
251 #model.add(Conv2D(256, (3, 3), padding="same"))
252 #model.add(Activation("relu"))
253 model.add(MaxPooling2D(pool_size=(2, 2)))
254
255 #model.add(Conv2D(512,(3,3),padding="same"))
```

```

256 #model.add(Activation("relu"))
257 #model.add(Conv2D(512, (3, 3), padding="same"))
258 #model.add(Activation("relu"))
259 #model.add(Conv2D(512, (3, 3), padding="same"))
260 #model.add(Activation('relu'))
261 #model.add(Conv2D(512, (3, 3), padding="same"))
262 #model.add(Activation("relu"))
263 #model.add(MaxPooling2D(pool_size=(2, 2)))
264
265 model.add(Flatten())
266 #model.add(Dense(4096))
267 #model.add(Activation("relu"))
268 model.add(Dense(512))
269 model.add(Activation("relu"))
270 model.add(Dense(128))
271 model.add(Activation("relu"))
272
273 model.add(Dense(classes))
274 model.add(Activation("softmax"))
275
276 print(model.summary())
277
278 # train the model using SGD
279 print("[INFO] training network...")
280 sgd = keras.optimizers.SGD(lr=Learning_Rate, decay=Decay_Rate, momentum=
    Momentum_Rate)
281 model.compile(loss="categorical_crossentropy", optimizer=sgd,
282     metrics=["accuracy"])
283
284 #Model check point
285 checkpoint = ModelCheckpoint(Weight_Save_Path, monitor="val_loss",
286     save_best_only=True, verbose=1)
287 callbacks = [checkpoint]
288
289 H = model.fit(aug.flow(trainX, trainY, batch_size=batch), validation_data = (valX,
290     valY), callbacks=callbacks,
291     epochs=ep, verbose=1)
292
293 # plot the training loss and accuracy
294 plt.style.use("ggplot")

```

```

295 plt.figure()
296 plt.plot(np.arange(0, ep), H.history["loss"], label="train_loss")
297 plt.plot(np.arange(0, ep), H.history["val_loss"], label="val_loss")
298 plt.plot(np.arange(0, ep), H.history["accuracy"], label="train_acc")
299 plt.plot(np.arange(0, ep), H.history["val_accuracy"], label="val_acc")
300 plt.title("Training Loss and Accuracy")
301 plt.xlabel("Epoch #")
302 plt.ylabel("Loss/Accuracy")
303 plt.legend()
304 plt.show()
305
306 """## 4. Model testing"""
307
308 #Model Testing Section
309 -----
310 import pandas as pd
311
312 model = load_model(Weight_Save_Path)
313 batch=BATCH_SIZE
314
315 # define label for each class
316 labelNames = [ "Sold", "Maybe", "Notbuy"]
317
318 print("[INFO] evaluation testing for training data ...")
319 # run prediction for training data
320 predictions = model.predict(trainX, batch_size=batch)
321 # print(predictions) # for debugging purpose
322 # print(trainY) # for debugging purpose
323 # dsisplay the performance
324 print(classification_report(trainY.argmax(axis=1),
325                             predictions.argmax(axis=1), target_names=labelNames))
326
327 # generate report as a dataframe
328 clsf_report = pd.DataFrame(classification_report(trainY.argmax(axis=1),
329                                                 predictions.argmax(axis=1),
330                                                 target_names=labelNames,
331                                                 output_dict=True)).transpose()
332
333 # write the report to csv file
334 clsf_report.to_csv(train_report_path, index= True)

```



```
335 print("[INFO] evaluation testing for test data ...")
336 # run prediction for testing data
337 predictions = model.predict(testX, batch_size=batch)
338 # dsiplay the performance
339 print(classification_report(testY.argmax(axis=1),
340     predictions.argmax(axis=1),target_names=labelNames))
341
342 # generate report as a dataframe
343 clsf_report = pd.DataFrame(classification_report(testY.argmax(axis=1),
344     predictions.argmax(axis=1),
345     target_names=labelNames,
346     output_dict=True)).transpose()
347
348 # write the report to csv file
349 clsf_report.to_csv(test_report_path, index= True)
350
351 print("[INFO] evaluation testing for validating data ...")
352 # run prediction for training data
353 predictions = model.predict(valX, batch_size=batch)
354 # dsiplay the performance
355 print(classification_report(valY.argmax(axis=1),
356     predictions.argmax(axis=1),target_names=labelNames))
357
358 # generate report as a dataframe
359 clsf_report = pd.DataFrame(classification_report(valY.argmax(axis=1),
360     predictions.argmax(axis=1),
361     target_names=labelNames,
362     output_dict=True)).transpose()
363
364 # write the report to csv file
365 clsf_report.to_csv(val_report_path, index= True)
```

BIOGRAPHY

Name Niracha Chaiwong

Education 2020: Bachelor of Engineering
(Electronics and Communication Engineering)
Sirindhorn International Institute of Technology
Thammasat University

