ILKOM Jurnal Ilmiah Vol. 13, No. 3, December 2021, pp. 226-234
Accredited 2nd by RISTEKBRIN No. 200/M/KPT/2020; E-ISSN 2548-7779 | P-ISSN 2087-1716

Research Article Open Access (CC–BY-SA)

Classification model of ‘toraja’ arabica coffee fruit ripeness levels using convolution neural network approach

Aryo Michael [a,1,*]**; Melki Garonga**[a,2]**;**
[a] *Program Studi Teknik Informarika UKI Toraja, Jl. Nusantara No. 12, Tana Toraja 91811, Indonesia*
[1] *aryomichael@ukitoraja.ac.id;* [2]*melkigaronga@ukitoraja.ac.id;*
** Corresponding author*

Article history: Received June 01, 2021; Revised July 28, 2021; Accepted July 29, 2021; Available online August 08, 2021

Abstract

The purpose of this study is to design a CNN deep learning algorithm model that can classify the ripeness level of Arabica coffee fruits based on image, the resulting model can be applied to a coffee bean sorting device based on artificial intelligence so that issues regarding to quality standards in the sorting process of Arabica coffee fruit can be avoided. This could lead to the improvement of the quality of Arabica Toraja coffee products. The research began from the collection of data in the form of raw Arabica Toraja coffee fruit images, 4000 images with 4 categories: unripe, semi ripe, perfectly ripe and overripe. CNN basic architecture was created using images with a size of 128x128 pixels, 4 convolution layers using 3x3 filters opening 32, 64, 128, and 256 with ReLU activation, followed by a poll layer with a 2x2 filter. The full connected layer used 2 hidden layers with dropout layers. The simulation model was conducted with 5-fold cross-validation method using epoch 100, 'adam' optimization algorithm with a learning rate of 0.0001, and batch size of 10. The successfulness of a model was measured based on the calculation of the confusion matrix. The test results showed that the accuracy rate of the third model using a combination of maxpolling and averagepolling performed best with an introduction accuracy of 98.75%, the first model used maxpolling with an accuracy of 98.25% while the lowest accuracy was on the second model used averagepolling with 97.75%.

Keywords: Deep Learning; Convolution Neural Network; Image Processing, Classification, Arabica Coffee

Introduction

Arabica coffee is one species of coffee that grows in Indonesia, especially in Toraja which is well known as one of the Arabica coffee producing areas in South Sulawesi. Coffee cultivation and processing technology plays a very important role in determining the quality of coffee, starting from the seeds selection, the planting process, harvesting methods, and post-harvest processing. In terms of post-harvest management, various activities are carried out to produce coffee beans that meet with the standards ranging from sorting coffee cherries to becoming coffee bean products that are ready to be marketed. Currently, the process of harvesting coffee cherries is carried out by smallholder plantations in both regions Tana Toraja and North Toraja. They harvest the cherries manually by directly picking coffee cherries from the tree. After that, a sorting process is carried out to separate the ripe coffee cherries. The last process takes a long time because it has many obstacles due to the subjective nature of the selection of coffee cherries or other factors such as the visual condition of the person doing the sorting. These could impact on the decline of the quality of the coffee produced. Technological developments, especially in the field of artificial intelligence (AI) in various domains make it possible to overcome these problems. The combination of machines combined with AI algorithms, makes machines that previously had no knowledge into intelligent machines.

In terms of determining the ripeness level of coffee cherries, it is usually identified from the change in color of the fruit. Arabica coffee cherries are usually green when it is unripe, slightly yellowish to reddish when it is semi ripe and bright red to dark red when ripe. The coffee fruit data is processed by image processing techniques through a feature extraction process to get data on the characteristics of the coffee fruits, then the machine learning algorithm will study these features for the learning process, then the machine learning algorithm performs grouping of the data that has been studied previously. Various studies have been conducted to determine the ripeness of coffee cherries. Research conducted by H. Syahputra et al [1] used the mean color moment features, standard deviation and color skewness of HSV coffee fruit images to see the characteristics of coffee fruit ripeness. Research by Widyaningsih et al. [2] compared the performance of GLCM feature extraction combined with fuzzy logic and K-Nearest Neighbor algorithms on coffee maturity classification where the results showed that the K-NN and GLCM algorithms produced better accuracy. Research conducted by E.Rachmawanto et al [3] applied the HSV image feature extraction and classification process using the K-NN algorithm with the highest accuracy of 93.33% at K=1 and K=3 with an accuracy value of 96.67%. Research conducted by Pulungan et al [4] applied color feature extraction and learning vector quantization (LVQ) artificial neural networks to identify coffee fruit maturity, the application of this algorithm resulted in 100% accuracy. Research by Muhammad Rioarda Irfa'I et al [5] conducted a classification of coffee fruit ripeness using HSV imagery and fuzzy c-means algorithm with 75% and 25% correctly recognized data. One of the obstacles in traditional machine learning is the selection of features to be used in the classification process. Selection of the right features will result in good performance, but the selection of inappropriate features will result in poor performance, so it takes special abilities for the researchers to see good features to be used in the classification process.

One branch of machine learning that is currently developing is deep learning. Deep learning algorithms have a very unique ability, namely an ability that can perform an automatic extraction, where the process of feature extraction and classification using learning is carried out by the algorithm itself [6]. The deep learning algorithm commonly used in image processing is Convolution Neural Network (CNN). CNN is an artificial neural network that is used to process data in the form of a matrix, by utilizing convolution operations [7]. In image processing, CNN features learning through a series of processes at the convolution layer. For this reason, CNNs do not require a special function extraction process and CNNs usually only require basic pre-processing to normalize the data. Based on this, in this study, research will be carried out to model the CNN architecture to classify the ripeness level of coffee cherries, so that the resulting model can later be applied to a device or machine for sorting coffee beans based on artificial intelligence.

Method

This research was conducted in a few stages which was described in following **Figure 1.**

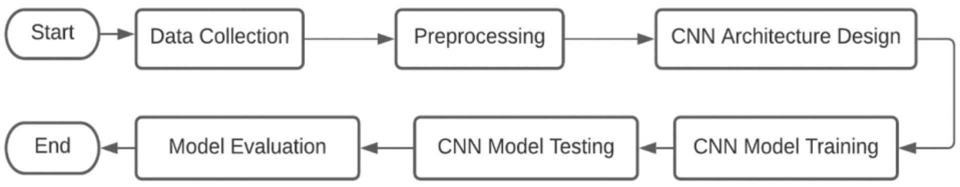


Figure 1. Reserch stages

A. Data Collection

Data collection aimed to obtain data used for the research. The data was obtained by taking images of Toraja Arabica coffee cherries in Arabica coffee producing areas in North Toraja and Tana Toraja which consisted of 4 levels of coffee fruit maturity, namely Arabica coffee cherries that were unripe, semiripe, perfectly ripe and overripe.

B. Preprocessing

At the preprocessing stage there were several simple image processing techniques carried out before classifying objects with CNN. The use of the OpenCV library was to help the process on the google collaborative. The initial process carried out was to equalize the image color space into an RGB color space, then equalize the size of the entire image dataset to a size of 128x128 pixels. The next process was image normalization by transforming all pixels in the image into 0-1 intervals. This process was conducted by dividing the pixel value in each image layer by 255. Labeling using one hot encoding was done to convert the categorical data into numerical form so that it can be processed on CNN.

C. CNN Architecture Design

The next stage was the CNN architecture design which aims to create a CNN architectural model that was employed to train the algorithm in recognizing objects. The CNN architecture consists of 3 (three) parts, namely the convolution layer, the subsampling layer/polling layer and the fully connected layer [6],[8],[9].

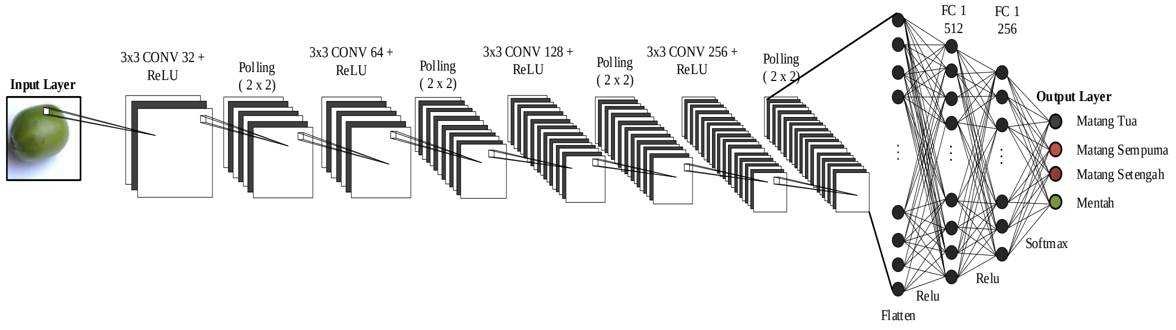


Figure 2. Suggested CNN Architecture Design

The proposed CNN architecture is presented in **Figure 2**. The proposed architecture consists of an input layer which was a color image (RGB) with a size of 128 x 128 pixels. The Learning feature layer consists of a combination of 4 convolution and polling layers: in the convolution layer the filter size used was 3x3 with a 1x1 shift (stride) and the number of filters in each layer was different, namely the first convolution layer consisting of 32 filters, the second convolution layer with 64 filters, the third convolution layer with 128 filters and the fourth convolution layer with 256 filters. Each convolution layer used the *ReLU* activation function and the polling would use a filter size of 2x2. The Fully connected layer consists of 2 (two) hidden layers with a dropout layer and the layer was the number of classifications. In this research there were 4 classifications with a *softmax* activation function. In this study, an experiment will be conducted by changing the type of polling layer used. The first model (Model1) used *MaxPolling*, the second model (Model2) used *AveragePolling* while the third model (Model3) used *Maxpolling* in the first 2 polling layers and Average Polling in the next polling layer.

D. CNN Model Simulation

The next stage is model simulation to obtain the best accuracy value for classifying Arabica coffee fruit ripeness. In model training, the K-fold Cross Validation technique used is a statistical method to evaluate and to compare the performance of the algorithm by dividing the data into a number of partitions according to the value of K[10],[11],[12]. In the first iteration, the first fold was used as validation data and the rest was used as simulation data. In the second iteration, the second fold was used as validation data. This process repeated until each fold had been used as validation data. Illustration of 5-Fold Cross Validation shown in **Figure 3**.

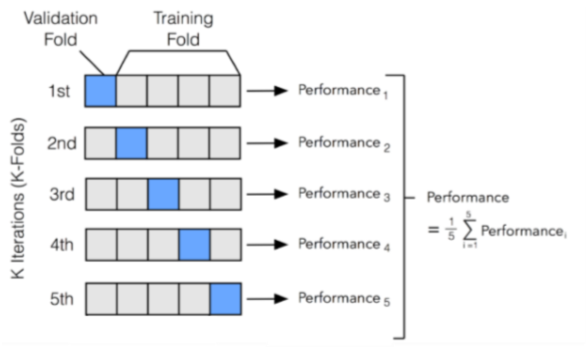


Figure 3. Illustration of 5-Fold Cross Validation

Determination of simulation parameters was very important, this greatly affected the performance of the model designed. Parameter determination included determining the value of learning rate, optimization method for changing parameters, number of epochs and batch size. The model training was carried out with several parameters on CNN, namely: "adams" optimization algorithm, learning rate = 0.0001, batch size = 10 and epoch = 100.

E. Testing and Evaluating Model

Stages of testing were carried out to determine the performance of the designed model. The prepared dataset was tested with the previously simulated model. The test was carried out using the fold which had the highest accuracy in each model. One of the methods used to measure the performance of a classification model with more than 2 classes (multiclass) was the confusion matrix. Confusion matrix can be used to help calculate accuracy, precision, recall and f1-score[13],[14],[15]. To calculate accuracy, it is shown in equation (1).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (1)$$

For the case of classification that employed multiclass precision calculations, recall and f1-score were carried out independently in each class. To calculate precision, recall and f1-score, equation (2), equation (3) and equation (4) were used.

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

$$f1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

Results and Discussion

A. Data Collection and Dataset Sharing

Image data retrieval was conducted using a smartphone camera with a resolution of 5MP. The image cutting process was carried out using the help of the *irfanView* application to reproduce coffee fruit image samples shown in **Figure 4**. Each level of coffee fruit ripeness collected 1000 images so that the total amount of image data collected was 4000 coffee images shown in **Table 1**. The images that had been collected and grouped by class were then divided into 2 parts, namely as a simulation dataset and a dataset testing.

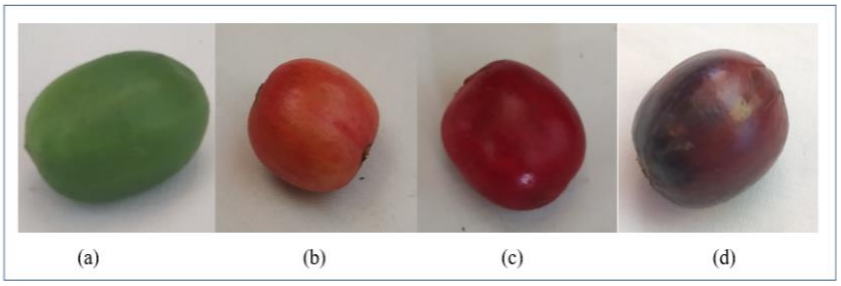


Figure 4. Arabica Fruit Images (a) unripe (b) semi ripe (c) Perfectly Ripe (d) overripe

Table 1. Dataset Sharing

Classification	Number of Dataset	
	Simulation	***Testing***
Unripe	800	200
Semi ripe	800	200
Perfectly ripe	800	200
Over ripe	800	200

B. Preprocessing

Image data retrieval was conducted using a smartphone camera with a resolution of 5MP. The image cutting process was carried out using the help of the *irfanView* application to reproduce coffee fruit image samples. Each level of coffee fruit ripeness collected 1000 images so that the total amount of image data collected was 4000 coffee images. The images that the preprocessing stage aimed to prepare Arabica coffee images which would be used as input in the classification process by the CNN algorithm. The image dataset that had been uploaded to Google Drive was then loaded into the Google Collaboratory application, then the image color space was equalized to RGB, the image size was equalized to 128x128 pixels which was later inputted to CNN for both model simulation and model testing. The process continued with the image normalization process by transforming all pixels in the image into 0 - 1 intervals by dividing the pixel value of each layer by 255 which was the highest pixel value in the RGB image. Furthermore, labeling of the dataset classes was carried out using the one hot encoding technique. **Figure 5** shows the preprocessed image dataset result.

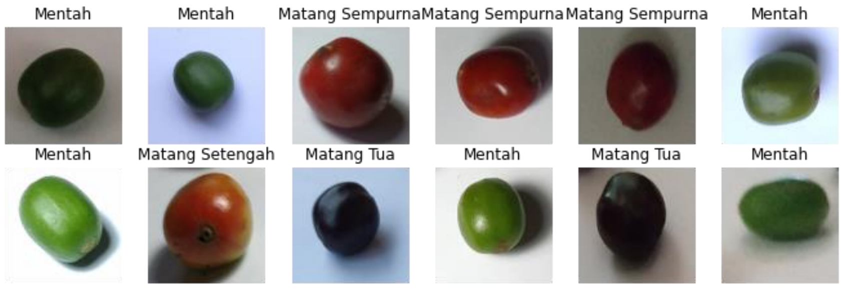
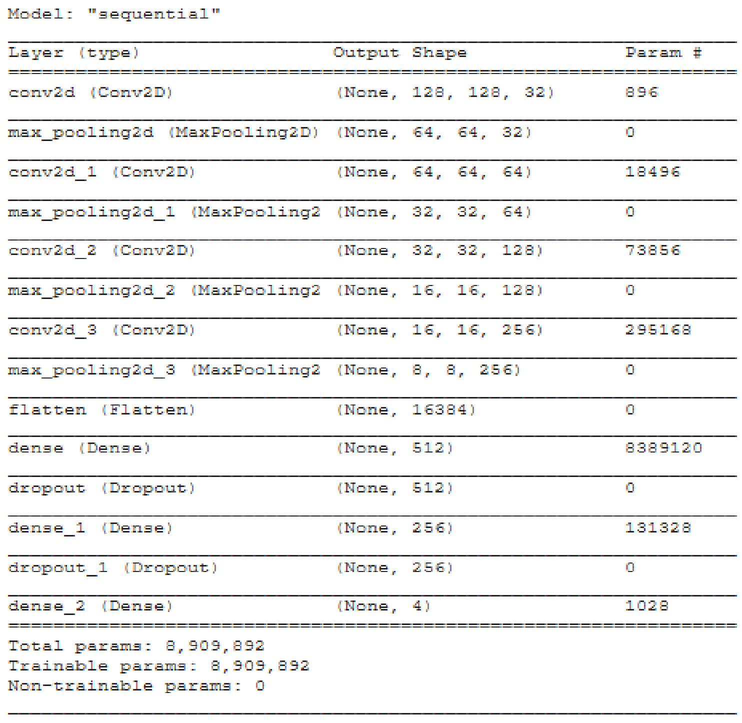


Figure 5. Image of Preprocessing Result

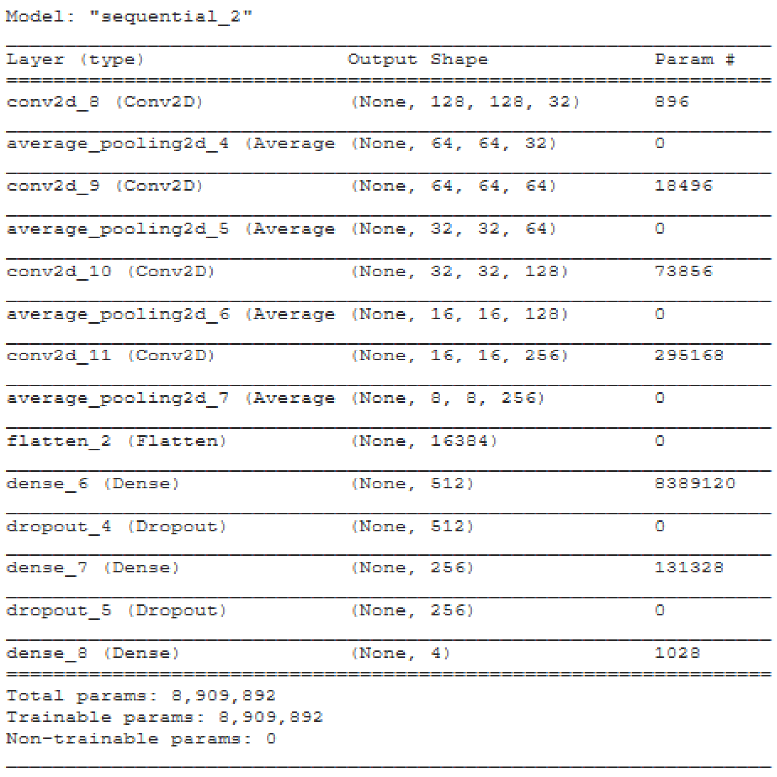
C. CNN Architecture Design

The CNN model that has been designed was then implemented in python in the google collaborative application using the hardware library and *tensorflow*. **Figure 6**, **Figure 7** and **Figure 8** were a summary of the models created using *the hardware* and *tensoflow* libraries.



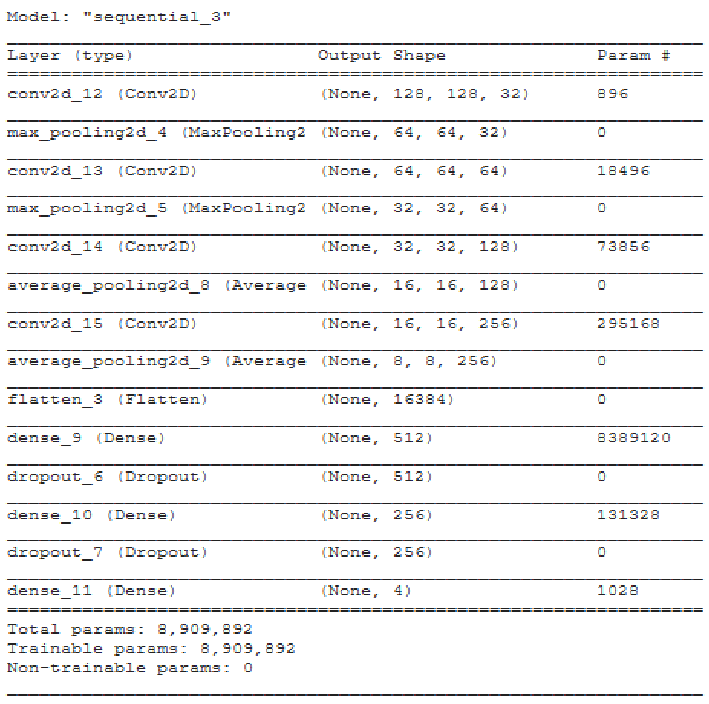
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 512)               8389120
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028
=================================================================
Total params: 8,909,892
Trainable params: 8,909,892
Non-trainable params: 0
_________________________________________________________________
```

Figure 6. Summary of Model1



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 128, 128, 32)      896
_________________________________________________________________
average_pooling2d_4 (Average (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
average_pooling2d_5 (Average (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 32, 32, 128)       73856
_________________________________________________________________
average_pooling2d_6 (Average (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 256)       295168
_________________________________________________________________
average_pooling2d_7 (Average (None, 8, 8, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 16384)             0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               8389120
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 4)                 1028
=================================================================
Total params: 8,909,892
Trainable params: 8,909,892
Non-trainable params: 0
_________________________________________________________________
```

Figure 7. Summary Model2



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 128, 128, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 64, 64, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 32, 32, 128)       73856
_________________________________________________________________
average_pooling2d_8 (Average (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 16, 16, 256)       295168
_________________________________________________________________
average_pooling2d_9 (Average (None, 8, 8, 256)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 16384)             0
_________________________________________________________________
dense_9 (Dense)              (None, 512)               8389120
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 1028
=================================================================
Total params: 8,909,892
Trainable params: 8,909,892
Non-trainable params: 0
_________________________________________________________________
```

Figure 8. Summary of Model3

D. Model Simulation

The simulation process for each model used 100 epoch parameters, the "adams" algorithm with learning rate = 0.0001, batch size = 10 using 5-fold cross validation. The results of each fold model simulation were saved in h5 format. From each model, the best fold accuracy would be taken and used as a model for the testing process. **Figure 9, Figure 10** and **Figure 11** show the training graphs and model validation. **Table 2** The CNN Model Training resulted in the highest accuracy of Model1 of 99.531 with a loss of 0.018 in the 2nd fold. Model2 had an accuracy of 99.375 with a loss of 0.016 in the 2nd fold and the highest accuracy in Model3 is 99.351 and a loss of 0.11 in the 2nd fold.

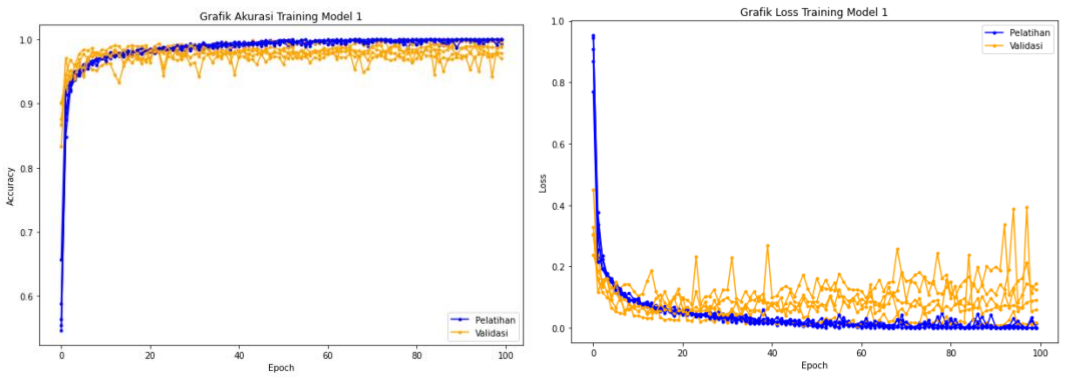


Figure 9. Accuracy Graph and Simulation Lost of Model1 with *batch size*=10

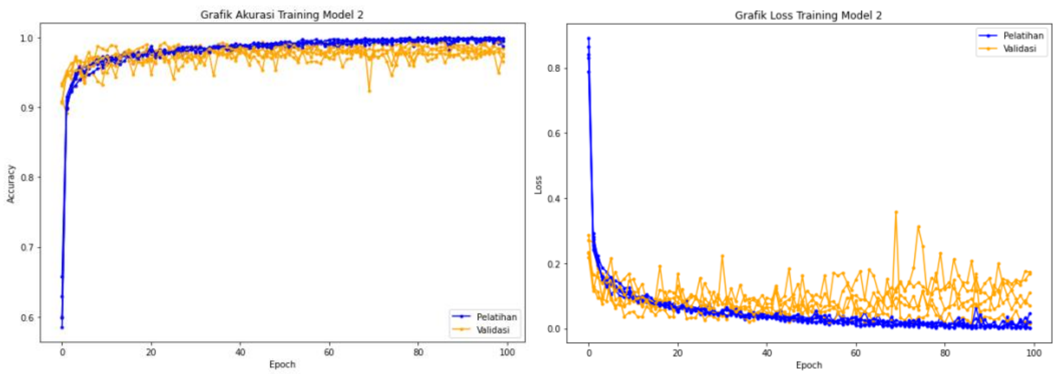


Figure 10. Accuracy Graph and Simulation Lost of Model2 *batch size*=10

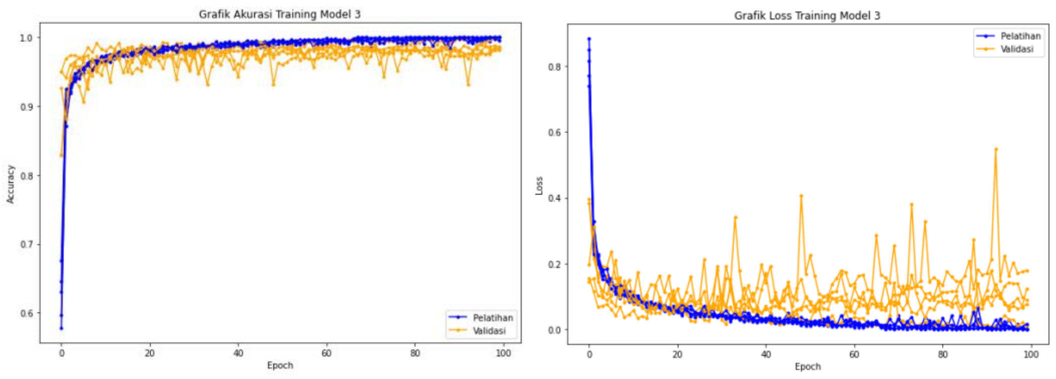


Figure 11. Accuracy Graph and Simulation Lost of Model3 *batch size*=10

Table 2. Simulation Result of CNN Model

Model	Fold 1		Fold 2		Fold 3		Fold 4		Fold 5	
	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
Model 1	97,031	0,146	99,531	0,018	98,750	0,091	97,812	0,124	98,906	0,059
Model 2	97,344	0,168	99,375	0,016	98,438	0,070	96,562	0,173	97,656	0,108
Model 3	98,281	0,180	99,531	0,011	98,438	0,124	98,125	0,089	98,594	0,077

E. Testing and Evaluating Models

Model testing was carried out using the results of the model simulation which performed best. Tests were carried out on the test data that had been prepared, 800 images of coffee cherries. **Figure 12** is a confusion matrix testing of each model.

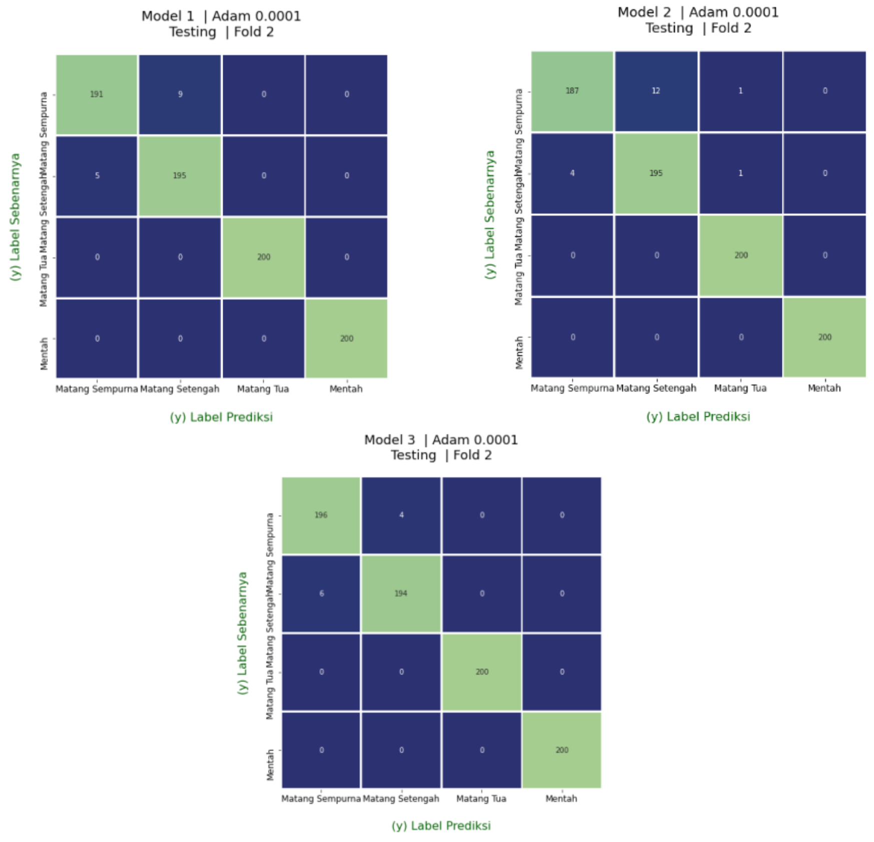


Figure 12. *Confusion Matrix* of Model Testing Result

From the testing process, the accuracy of Model1 was 98.25%, Model2 was 97.75%, and Model3 was 98.75%. The use of *maxpolling* in the pooling layer provided a sharper image because it focused on the maximum feature value consequently there were some important features in objects that had a smaller value would be ignored, on the contrary the use of average polling would still maintain the essence of the existing features because it took the average value of the features obtained from the previous process, the use of a combination of *maxpolling* and *averagepolling* still paid attention to these two things so as to provide more information about the important features of the identified coffee fruit image. The performance of each model which was calculated using equation (2), equation (3) and equation (4) is presented in **Table 3, Table 4** and **Table 5.**

Table 3. Testing Result of Model 1

Category	**Model1**			
	Precision	***Recall***	***f1-score***	***Accuracy***
Perfectly ripe	0,97	0,96	0,96	98,25%
Semi ripe	0,96	0,98	0,97	
Over ripe	1,00	1,00	1,00	
unripe	1,00	1,00	1,00	

Table 4. Testing Result of Model 2

Category	**Model2**			
	Precision	***Recall***	***f1-score***	***Accuracy***
Perfectly ripe	0,98	0,94	0,96	97,75%
Semi ripe	0,94	0,98	0,96	
Over ripe	0,99	1,00	1,00	
Unripe	1,00	1,00	1,00	

Table 5. Testing Result of Model 3

Category	**Model3**			
	Precision	***Recall***	***f1-score***	***Accuracy***
Perfectly ripe	0,97	0,98	0,97	98,75%
Semi ripe	0,98	0,97	0,97	
Over ripe	1,00	1,00	1,00	
Unripe	1,00	1,00	1,00	

Conclusion

The CNN Architecture Model for classifying the ripeness level of coffee cherries has been successfully implemented using the hard library and *tensorflow*. From the tests carried out on the proposed CNN Architecture, Model3 employing a combination of *maxpolling* and *averagepolling* has the best performance with a recognition accuracy of 98.75%, Model1 uses *maxpolling* with an accuracy of 98.25% while the lowest accuracy is in Model2 with an accuracy of 97.75%. As a continuation of this research, model development would be carried out, especially on the use of the input image color space and the use of other classification algorithms in *fully connected.*

Acknowledgement

The author would like to thank UKI Toraja for giving the author the opportunity to carry out this research, as well as all parties who helped so that this research can be carried out.

References

[1] H. Syahputra, F. Arnia, and K. Munadi, “Karakterisasi Kematangan Buah Kopi Berdasarkan Warna Kulit Kopi Menggunakan Histogram dan Momen Warna,” *J. Nas. Tek. Elektro*, vol. 8, no. 1, p. 42, 2019, doi: 10.25077/jnte.v8n1.615.2019.

[2] Widyaningsih, I. I. Tritosa, and N. C. Kumalasari, “Perbandingan Klasifikasi Tingkat Kematangan Buah Kopi Menggunakan Metode Fuzzy Logic Dan K-Nearest Neighbor Dengan Ekstraksi Ciri Gray Level Co-Occurrence Matrix Comparison of Coffee Cherries Ripeness Using Fuzzy Logic and K- Nearest Neighbor Method With,” vol. 7, no. 2, pp. 4060–4073, 2020.

[3] E. H. Rachmawanto and A. Salam, “Pengukuran Tingkat Kematangan Kopi Robusta menggunakan Algoritma K-Nearest Neighbor,” in *Prosiding SENDI_U*, 2018, pp. 978–979.

[4] W. A. Pulungan, Y. Mulyani, and W. E. Sulistiono, “Identifikasi Kematangan Buah Kopi Menggunakan Jaringan Syaraf Tiruan Learning Vector Quantization,” *Barometer*, vol. 4, no. 2, p. 217, 2019, doi: 10.35261/barometer.v4i2.1834.

[5] M. Rioarda, B. Fatkhurrozi, and I. Setyowati, “Klasifikasi Tingkat Kematangan Buah Kopi Menggunakan Algoritma Fuzzy C – Means,” *THETA OMEGA J. Electr. Eng. Comput. Inf. Technol.*, vol. 2, no. 1, 2021,

[Online]. Available: https://jurnal.untidar.ac.id/index.php/thetaomega/article/view/3913/1895.
[6] I. W. Suartika, A. Y. Wijaya, and R. Soelaiman, "Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101," *J. Tek. ITS*, vol. 5, no. 1, pp. A65–A69, 2016, [Online]. Available: ejurnal.its.ac.id/index.php/teknik/article/viewFile/15696/2553.
[7] A. M. Rizki and N. Marina, "Klasifikasi Kerusakan Bangunan Sekolah Menggunakan Metode Convolutional Neural Network Dengan Pre-Trained Model Vgg-16," *J. Ilm. Teknol. dan Rekayasa*, vol. 24, no. 3, pp. 197–206, 2019, doi: 10.35760/tr.2019.v24i3.2396.
[8] L. Alzubaidi *et al.*, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021.
[9] A. Taner, Y. B. Öztekin, and H. Duran, "Performance Analysis of Deep Learning CNN Models for Variety Classification in Hazelnut," *Sustainability*, vol. 13, no. 12, p. 6527, 2021, doi: 10.3390/su13126527.
[10] A. Peryanto, A. Yudhana, and R. Umar, "Klasifikasi Citra Menggunakan Convolutional Neural Network dan K Fold Cross Validation," *J. Appl. Informatics Comput.*, vol. 4, no. 1, pp. 45–51, 2020, doi: 10.30871/jaic.v4i1.2017.
[11] S. Ilahiyah and A. Nilogiri, "Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," *JUSTINDO (Jurnal Sist. dan Teknol. Inf. Indones.*, vol. 3, no. 2, pp. 49–56, 2018.
[12] H. Azis, P. Purnawansyah, F. Fattah, and I. P. Putri, "Performa Klasifikasi K-NN dan Cross Validation Pada Data Pasien Pengidap Penyakit Jantung," *Ilk. J. Ilm.*, vol. 12, no. 2, pp. 81–86, 2020, doi: 10.33096/ilkom.v12i2.507.81-86.
[13] Y. Baştanlar and M. Ozuysal, *Introduction to Machine Learning Second Edition*, vol. 1107. 2014.
[14] A. Tharwat, "Classification Assessment Methods," *Appl. Comput. Informatics*, vol. 17, no. 1, pp. 168–192, 2018, doi: 10.1016/j.aci.2018.08.003.
[15] I. M. Erwin, R. Risnandar, E. Prakarsa, and B. Sugiarto, "Kayu7net: Identifikasi dan Evaluasi F-Measure Citra Kayu berbasis Deep Convolutional Neural Network (DCNN)," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 6, p. 1089, 2020, doi: 10.25126/jtiik.2020712663.