Detection of Varian Beans Coffee Using Pre-Train Model Resnet50V2 Convolutional Neural Networks

Pradita Eko Prasetyo Utomo[*1),] Benedika Ferdian Hutabarat [1)], Amanda Stiawan [1)], M, Taufik Hidayat [1)]

*Corresponding author
*ORCHID IDs: https://orcid.org/0009-0008-4787-4609

[1)]*Department of Information System, Universitas Jambi, Indonesia*
**email: pradita.eko@unja.ac.id*

Abstract

Farmers, coffee experts, coffee observers or agricultural experts (researchers) differentiate varieties or types of coffee by sight and based on the knowledge they have. Differentiating types or varieties of coffee is based on differences in color, shape and texture. However, not all farmers, researchers or coffee observers can recognize varieties or types of coffee by just looking at raw coffee beans (green beans), coffee berries dan coffee leaves. So this is very possible, there could be errors in recognizing varieties or types of coffee if you don't have knowledge about coffee. Identification of types of coffee is difficult to differentiate with the naked eye so special skills are required. One method that can be used to identify types of coffee is to use digital image processing. Convolutional Neural Network is a type of deep learning that is often used in image processing. This research classifies varians of beans coffee such as arabica or robusta. For detection modeling, the proposed pre-trained model Restnet50V2 is used. From the results obtained with accuracy reaching 99.50%, it can be concluded that the model created can perform classification well. With a loss value of only 0.0316, which is a very small loss value.

Keyword: CNN, Resnet50V2, beans of coffe, light roast, dark roast, roasted, maturity, deep learning

I. Introduction

Coffee is one of the plantation crops which is a source of income for the people and can also be a source of increasing the country's foreign exchange through the export of raw coffee beans and processed coffee beans (Nugraha et al, 2018). Indonesia is the fourth largest coffee bean producing country in the world after Brazil, Vietnam and Colombia with an average production of around 700 thousand tons per year or around 9% of world coffee production. Based on this, according to the Director General of Agro Industry, Ministry of Industry, domestic coffee bean processing must continue to be improved (Prastyaningsih et al, 2020). Indonesia has the opportunity to develop the coffee processing industry, because apart from having a large market, it is also supported by potential raw materials. Therefore, strategic efforts are needed, such as downstreaming in order to increase added value and increase production capacity (Ministry of Industry, 2019). Coffee can be differentiated based on its type, namely Robusta Coffee and Arabica Coffee. Arabica coffee has various varieties based on the region of origin or is known as single origin coffee. Each coffee variety has a different price depending on the type of variety. However, not all farmers and coffee shop owners are able to recognize coffee varieties by just looking at the green beans and roasting. So, errors can occur in recognizing coffee varieties if the coffee shop owner does not have knowledge about coffee. This can be overcome by modeling that can identify Robusta and Arabica coffee varieties so that they can be used as a second opinion to identify coffee bean varieties. One of the methods used is the imaging method (Nugroho et al, 2020). The use of digital images has been widely used to identify objects, from small to large

(Syahputra et al, 2019). The comparison between coffee beans and other coffee beans can be identified from the texture, color and weight of the coffee beans. However, coffee beans are often found that have the same characteristics as other coffee beans. The similarities between coffee beans generally lie in the texture and weight of the 2 coffee beans. For this reason, some ordinary people find it difficult to identify certain types of coffee beans because in terms of texture and weight the coffee beans are relatively almost the same (Kristanto, 2018). As we know, technological developments in the field of digital image processing have been very rapid, especially in the techniques used to classify types of coffee beans. It is hoped that this alternative technological assistance can help increase the effectiveness of the coffee bean classification process to make it better and more efficient. One of the technologies that utilizes digital image processing is deep learning. The Deep Learning method that currently has the most significant results in image processing is Convolutional Neural Network (CNN). This is because the CNN method tries to imitate the image recognition system in the human visual cortex so that it has the ability to process image information like humans (Maulana & Rochmawati, 2019). This research aims to build a coffee bean type modeling system using the CNN model with the Resnet50 pre-train method. The benefit of this research is that coffee farmers and sellers can easily differentiate between types of beans, especially Robusta and Arabica coffee.

II. Method

2.1 Coffee Varians

Coffee is one of the plantation commodities that has quite high economic value, among other plantation crops and plays an important role as a source of foreign exchange for the country. Coffee not only plays an important role as a source of foreign exchange, but is also a source of income for no less than one and a half million coffee farmers in Indonesia (Rahardjo, 2013).

Figure 1. Types of Coffee

The famous types of coffee in Indonesia are robusta (Coffea canephora) and arabica (Coffea arabica L.). According to the Ministry of Agriculture (2017), in 2016 Indonesia's coffee production reached 693.3 thousand tons. Robusta coffee accounts for 81% of the total coffee production in Indonesia and the remainder is Arabica coffee. West Java is one of the largest Arabica coffee production centers in Indonesia with total production of up to 9.37 thousand tons per year.

2.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is developed based on the multilayer layer perceptron (MLP) which is designed or intended for processing two dimensional data in the form of images. CNN is a variant of the deep neural network because of its high network depth and is widely applied to complex image data (Rismiyati and Azhari, 2016). The image classification process can basically only use MLP, but one of the weaknesses of this MLP method is that it is not suitable because it cannot store spatial information from the mind data and each pixel is considered an independent or large feature, thus allowing to get unfavorable results (Juliansyah, 2019).

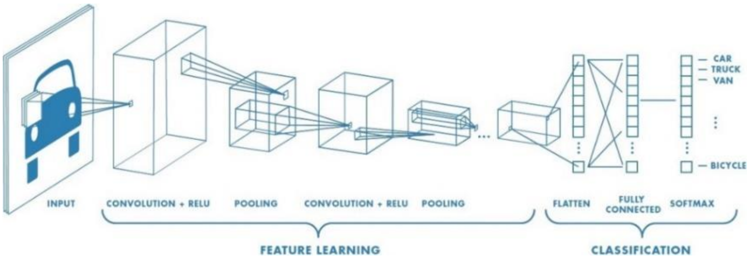
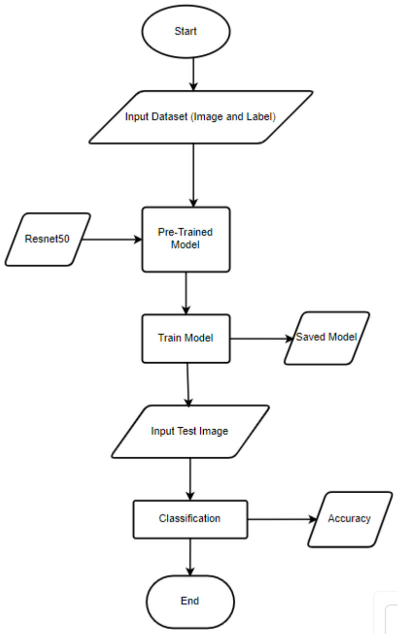


Figure 2. Architecture of CNN

The CNN method is the same as other neural network theories which are trained using the backpropagation algorithm. CNN is designed to recognize visual patterns directly from image pixels by minimizing preprocessing. CNN can recognize patterns with a wide variety, resistant to distortion and simple geometric transformations. The architecture of CNN is divided into major parts, namely the Feature Extraction Layer and the Fully Connected Layer. Each stage consists of three layers, namely the convolutional layer, the layer activation function, and the pooling layer, which can be seen in Figure 2 which is the CNN network architecture.

2.3 Research Design



III. Result and Discussion

3.1 Collection dataset

At this stage, data collection is carried out first, the data collection process is carried out by taking photos directly on coffee beans using a camera. The total data collected was 400 images consisting of images of arabica and robusta coffee beans, fruits and leaves. The following is the data obtained and will be used as secondary data in model training.


Figure 3 Arabika and robusta bean

3.2 Remove background

Then the background removal will be done on each image to leave the main object of each image in the form of coffee beans. This is so that in conducting training on the CNN model will read only the main object. This will improve accuracy because it can ignore other objects and leave a plain background. The main goal is to maximize the program's performance in analyzing image pixels. This process is important because background images that have random colors can affect the program's readability of the image pixels. Background removal aims to separate the coffee bean image from the backgound. This step is important to ensure that only the coffee bean image remains in the image, so that it can be processed further. In addition, the process of removing the background and changing it to white / black (monochrome), this is done to maximize the performance of the program in reading image pixels, because if the image uses a default background whose color is random it will affect the program's reading of the image pixels.

```
import os
import rembg

def remove_background(input_folder, output_folder):
    # Buat folder output jika belum ada
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Loop melalui setiap file gambar dalam folder input, hapus background, dan simpan
hasilnya di folder output
    file_list = os.listdir(input_folder)

    for filename in file_list:
        input_filepath = os.path.join(input_folder, filename)
        output_filepath = os.path.join(output_folder, filename)

        # Membuka file gambar
        with open(input_filepath, "rb") as img_file:
            # Menghapus latar belakang menggunakan Rembg
            output_img = rembg.remove(img_file.read())

        # Menyimpan gambar tanpa latar belakang di folder output
        with open(output_filepath, "wb") as output_file:
            output_file.write(output_img)

input_folder2 = r'D:\TUGAS_AKHIR\DATASET_PRIMER\ARABIKA_BEANS'
output_folder2 = r'D:\TUGAS_AKHIR\DATASET_PRIMER\A'

remove_background(input_folder2, output_folder2)
```

In the process of removing the background using the library that has been provided by python, namely rembg.The following image is the image before and after background removal:

Figure 4 After remove background

3.3 Split dataset

Dataset splitting aims to separate datasets as data train test and validation. Data is separated into 80% data train, 10% test data and 10% validation data.

```
import os
import shutil
import random

# Define path to original data directory
data_dir = r'D:\manda\data primer coba\rembg'
```

```
# Define path to directories for train, test, and validation data
train_dir = r'D:\manda\data primer coba\train'
test_dir = r'D:\manda\data primer coba\test'
val_dir = r'D:\manda\data primer coba\validation'

# Define labels (arabika and robusta)
labels = ["ARABIKA", "ROBUSTA"]

# Set the percentage of images to be used for validation
val_ratio = 0.1  # 10% for validation
test_ratio = 0.1  # 10% for testing
train_ratio = 1 - (val_ratio + test_ratio)  # 80% for training

# Create the train, test, and validation directories if they don't exist
for label in labels:
    train_label_dir = os.path.join(train_dir, label)
    test_label_dir = os.path.join(test_dir, label)
    val_label_dir = os.path.join(val_dir, label)

    os.makedirs(train_label_dir, exist_ok=True)
    os.makedirs(test_label_dir, exist_ok=True)
    os.makedirs(val_label_dir, exist_ok=True)

# Loop through each label
for label in labels:
    # Define the path to the original data directory for the current label
    data_label_dir = os.path.join(data_dir, label)

    # Get the list of image filenames in the data directory
    img_filenames = os.listdir(data_label_dir)

    # Shuffle the list of filenames
    random.shuffle(img_filenames)

    # Calculate the number of images to use for validation and testing
    num_val = int(len(img_filenames) * val_ratio)
    num_test = int(len(img_filenames) * test_ratio)

    # Copy images for validation
    for filename in img_filenames[:num_val]:
        src = os.path.join(data_label_dir, filename)
        dst = os.path.join(val_dir, label, filename)
        shutil.copyfile(src, dst)

    # Copy images for testing
    for filename in img_filenames[num_val:num_val + num_test]:
        src = os.path.join(data_label_dir, filename)
        dst = os.path.join(test_dir, label, filename)
        shutil.copyfile(src, dst)

    # Copy the remaining images for training
    for filename in img_filenames[num_val + num_test:]:
        src = os.path.join(data_label_dir, filename)
        dst = os.path.join(train_dir, label, filename)
        shutil.copyfile(src, dst)
```

The following is the result of the separation:

```
train length: 320   test length: 40   valid length: 40
```

Figure 5. Split dataset

3.4 Augmentation

Augmentation is the most important part of processing images in order to multiply the dataset by taking into account the aspect of shooting angle of view. Some image augmentation processes include rescale, rotation, shift, zoom, flip, and fill. This process aims to improve the quality and diversity of datasets, so that the resulting model can better recognize various variations of coffee beans. In this study, data preprocessing is carried out with augmentation techniques so that the computer will detect that the changed image is a different image. The augmentation process carried out is in the form of rotation range to 15, rescaling to 1/255, shearing with a scale of 0.2, doing horizontal flip then doing width and height shift with a range of 0.1. After the process is carried out, the variables that have been created previously will then be called again for the implementation process of the dataset, part train_generator as well as for other data. As well as for the results of the preprocessing can be seen in Figure 21. It uses a variety of augmentation techniques, including zoom change, rotation, horizontal flip, and other vertical playback. These augmented images will be saved in a new folder corresponding to their original class.

```
datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
)
```

Before augmentation:

After augmentation:

Figure 1. Augmentation image

3.5 Preprocessing

Preprocessing is done to adjust the image size, batch size, model class and shuffle of the dataset. In this stage, the images are resized to 224x224 pixels, as they have become a common standard in image processing for training convoluted neural network (CNN) models and images not in RGB color mode will be converted to RGB mode. The images that have been resized and processed are then saved in a new folder according to their original class.

```
#create image data generator
train_gen = datagen.flow_from_directory(
        os.path.join(base_dir, 'train', 'augmented'),
        target_size=img_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=True
)
```

The output generated from the preprocessing stage is:

```
test batch size: 1   test steps: 199
Found 1534 images belonging to 2 classes.
Found 199 images belonging to 2 classes.
Found 199 images belonging to 2 classes.
['ARABIKA', 'ROBUSTA']
```

Figure 6. Preprocessing dataset

3.6 Create base model using pre-train

Modeling was done using the ResNet50V2 pre-train model. The pretrain method that has been provided will be used in training with the new dataset owned. In the formation of the model, it is necessary to determine the input format on the model and determine the base model pretrain used.

```
base_model = ResNet50V2(include_top=False, weights='imagenet',
input_tensor=Input(shape=(224,224,3)))
```

3.7 Create model

Next, form a model using the previous base model by adding several layers as follows:

1. Base model layer, is a layer that uses a pretrain model that has been initialized previously as the basemodel.
2. GlobalAveragePooling2D Layer
3. Dense layer with activation "relu'
4. Dropout layer with rate "0."
5. Dense layer with 2 as the number of classes with activation "softmax"

Next is to compile the model using "Adam" optimization with a learning rate of "0.00005" with loss = "categorical_crossentropy" and metric = ["accuracy"]. The result of creating the model is as follows:

```
base_model.trainable = False

model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(rate=0.3),
        tf.keras.layers.Dense(2, activation='softmax')
       ])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00005),
loss='categorical_crossentropy', metrics=['accuracy'])
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50v2 (Functional)     (None, 7, 7, 2048)        23564800

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 23827330 (90.89 MB)
Trainable params: 262530 (1.00 MB)
Non-trainable params: 23564800 (89.89 MB)
```

Figure 7. Model

3.8 Epoch

After the formation of the model, it can be done Epoch or training on the model with datasets that have been preprocessed and augmented. Here are the summary results of the 20 epoch process:

```
Epoch 1/20
24/24 [==============================] - 156s 6s/step - loss: 0.5634 - accuracy: 0.7327 - val_loss: 0.2444 -
Epoch 2/20
24/24 [==============================] - 28s 1s/step - loss: 0.2933 - accuracy: 0.8807 - val_loss: 0.1587 - v
Epoch 3/20
24/24 [==============================] - 28s 1s/step - loss: 0.1966 - accuracy: 0.9218 - val_loss: 0.1215 - v
Epoch 4/20
24/24 [==============================] - 28s 1s/step - loss: 0.1529 - accuracy: 0.9465 - val_loss: 0.1203 - v
Epoch 5/20
24/24 [==============================] - 29s 1s/step - loss: 0.1333 - accuracy: 0.9505 - val_loss: 0.0814 - v
Epoch 6/20
24/24 [==============================] - 28s 1s/step - loss: 0.1299 - accuracy: 0.9550 - val_loss: 0.1105 - v
Epoch 7/20
24/24 [==============================] - 28s 1s/step - loss: 0.0992 - accuracy: 0.9602 - val_loss: 0.0612 - v
Epoch 8/20
24/24 [==============================] - 28s 1s/step - loss: 0.0971 - accuracy: 0.9674 - val_loss: 0.0614 - v
Epoch 9/20
24/24 [==============================] - 27s 1s/step - loss: 0.0914 - accuracy: 0.9668 - val_loss: 0.0769 - v
Epoch 10/20
24/24 [==============================] - 28s 1s/step - loss: 0.0802 - accuracy: 0.9713 - val_loss: 0.0732 - v
Epoch 11/20
24/24 [==============================] - 28s 1s/step - loss: 0.0777 - accuracy: 0.9720 - val_loss: 0.0599 - v
Epoch 12/20
24/24 [==============================] - 28s 1s/step - loss: 0.0768 - accuracy: 0.9733 - val_loss: 0.0505 - v
Epoch 13/20
24/24 [==============================] - 28s 1s/step - loss: 0.0622 - accuracy: 0.9824 - val_loss: 0.0596 - v
Epoch 14/20
24/24 [==============================] - 28s 1s/step - loss: 0.0529 - accuracy: 0.9844 - val_loss: 0.0602 - v
Epoch 15/20
24/24 [==============================] - 28s 1s/step - loss: 0.0599 - accuracy: 0.9791 - val_loss: 0.0317 - v
Epoch 16/20
24/24 [==============================] - 28s 1s/step - loss: 0.0657 - accuracy: 0.9785 - val_loss: 0.0276 - v
Epoch 17/20
24/24 [==============================] - 28s 1s/step - loss: 0.0593 - accuracy: 0.9824 - val_loss: 0.0710 - v
Epoch 18/20
24/24 [==============================] - 29s 1s/step - loss: 0.0658 - accuracy: 0.9785 - val_loss: 0.0632 - v
Epoch 19/20
24/24 [==============================] - 28s 1s/step - loss: 0.0530 - accuracy: 0.9831 - val_loss: 0.0516 - v
Epoch 20/20
24/24 [==============================] - 29s 1s/step - loss: 0.0491 - accuracy: 0.9831 - val_loss: 0.0414 - v
```

Figure 8. Epoch History

3.9 Evaluation of accuracy

Evaluation of accuracy is an evaluation carried out in seeing how well the model is able to make predictions or classifications of test data. Accuracy is how well the model classifies appropriately while loss value is how well it models existing data. The results of the evaluation can illustrate how well the model that has been developed in performing calibration.

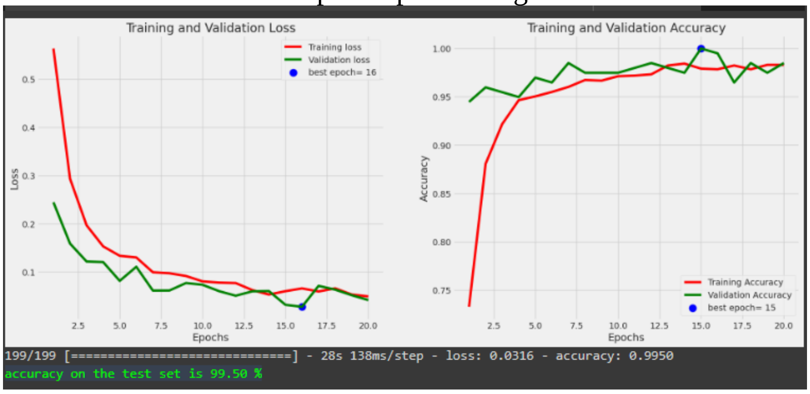


Figure 9. Evaluation of accuracy

From the results obtained with accuracy results reaching 99.50%, it can be concluded that the model made can perform well classification. With a loss value of only 0.0316, which is a very small loss value.

3.10 Confusion matrix

Confusion matrix aims to measure the error of the model against the test data. From visualization, it can be seen how well the ability of the model in classifying. How many errors occur can be seen in the confusion matrix visualization. The results of confusion matrix visualization can be seen as follows:

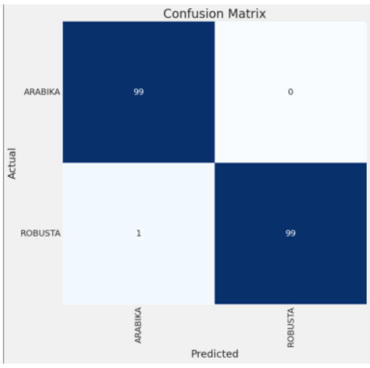


Figure 10. Confusion matrix

From the visualization, it can be seen that the model seen has only one image that has an error in determining the classification. So it can be concluded that the model made is correct that it has a very high auration as the results of previous training carried out.

	precision	recall	f1-score	support
ARABIKA	0.99	1.00	0.99	99
ROBUSTA	1.00	0.99	0.99	100
accuracy			0.99	199
macro avg	0.99	0.99	0.99	199
weighted avg	1.00	0.99	0.99	199

Figure 11. Classification report

From the statistics displayed with an accuracy that touches 0.99(99%), it can be concluded that the model developed can classify well and accurately from all test data.

3.11 Testing model

The test was carried out using several photos of green coffee beans with the number of experiments as much as 5 times with the following results:

Num	Image	Actual	Predicted	Confidence	True/false
1		Arabika	Arabika	Arabika:0.98926127 Robusta:0.01073873	True
2		Robusta	Robusta	Arabika:1.2693524e-29 Robusta:1.0000000e+00	True
3		Robusta	Robusta	Arabika:1.5760472e-26 Robusta:1.0000000e+00	True
4		Arabika	Robusta	Arabika:4.3721665e-21 Robusta:1.0000000e+00	False

IV. Conclusion

The classification model for varian beans coffee using the ResNet50 V2 Convolutional Neural Network (CNN) architecture has been proven to have very good accuracy of 99,50%.

References

H. Syahputra, F. Arnia, dan K. (2019). 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.

Juliansyah, L., Agustian, I dan Hadi, F., 2018. Deteksi Gangguan Ginjal Melaui Citra Digital Iris Mata Menggunakan Metode Convolutional Neural Network (CNN). Jurnal Amplifier Vol. 9 No. 1, Mei 2019 hal 1-9

Kristanto, Aditya. (2018). Image Processing Klasifikasi Biji Kopi Menggunaka Metode K-Nearest Neighbor. Yogyakarta : Universitas Kristen Duta Wacana.

Kementerian Perindustrian. (2019). “Industri Pengolahan Kopi Semakin Prospektif”. https://kemenperin.go.id/artikel/21117/IndustriPengolahan-Kopi-Semakin-Prospektif.

Maulana, F.F and Rochmawati, N (2019). Klasifikasi Citra Buah Menggunakan Convolutional Neural Network. JINACS: Volume 01 Nomor 02, 2019. Pp 104-108.

Nugroho, M. A. dan M. M. Sebatubun. (2020). KlasifikasiI Varietas Kopi Berdasarkan Green Bean Coffe Menggunakan Metode Machine Learning. Jurnal Of Information System Management Vol. 1: 1-5.

Nugraha, D. A. dan A. S. Wiguna. (2018). Seleksi Fitur Warna Citra Digital Biji Kopi Menggunakan Metode Principal Component Analysis. Journal of Computer, Information System, & Technology Management Vol. 3: 24-30.

Prastyaningsih, Y., A. Noor dan A. Supriyanto. (2020). Identifikasi Jenis Biji Kopi Menggunakan Ekstraksi Fitur Tekstur Berbasis Content Based Image Retrieval. ScientiCO : Computer Science and Informatics 3: 105-116

Rahardjo, Pudji. 2013. Kopi Panduan Budi Daya dan Pengolahan Kopi Arabika dan Robusta: Penebar Swadara

Rismiyati and SN. Azhari, “Convolutional Neural Network implementation for imagebased Salak sortation,” ICST (International Conference on Science and TechnologyComputer, 27-28 Oct. 2016 [Online]. Available:https://ieeexplore.ieee.org/document/7877351.