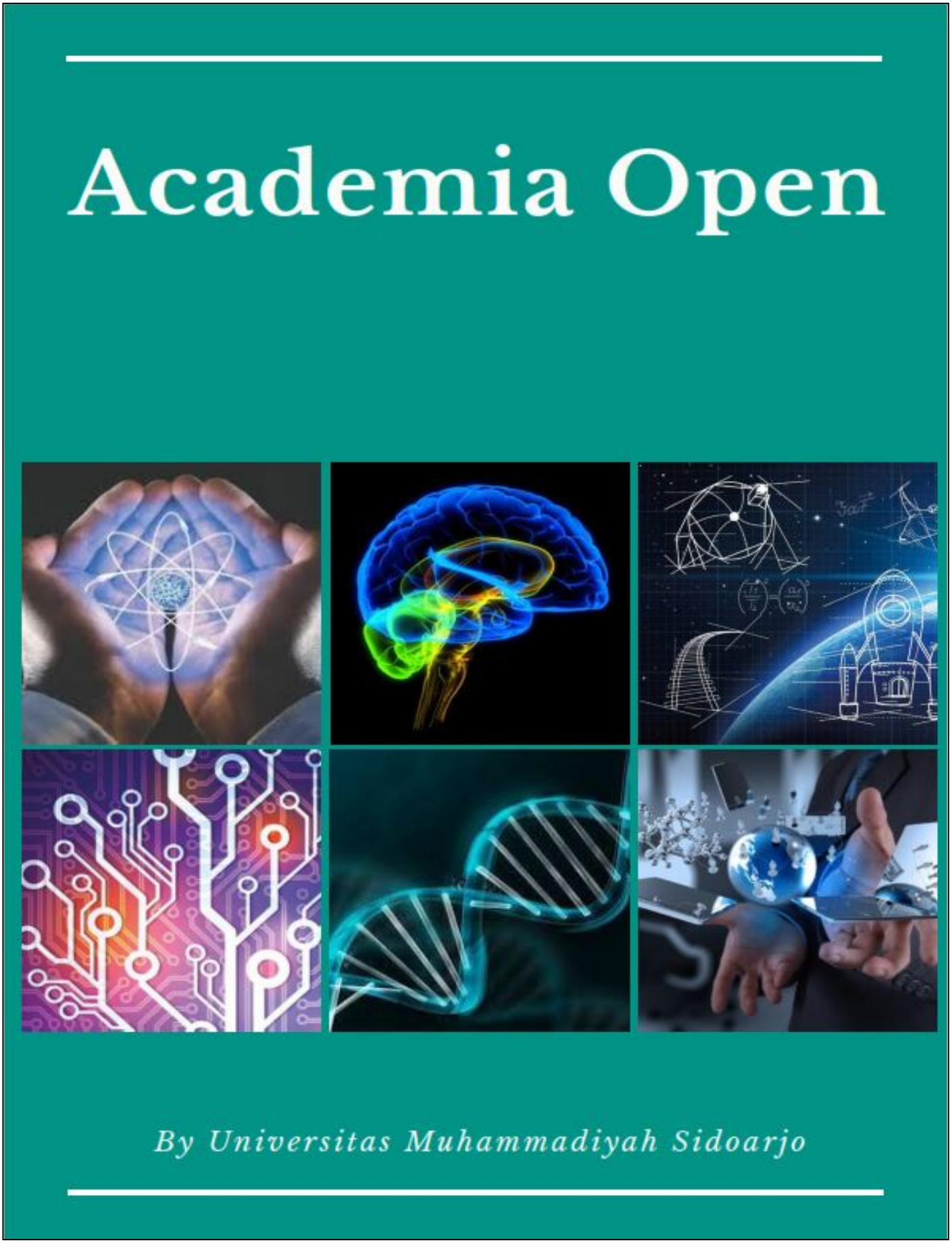
Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

Table Of Content

Journal Cover 2
Author[s] Statement 3
Editorial Team 4
Article information 5
Check this article update (crossmark) 5
Check this article impact 5
Cite this article 5
Title page 6
Article Title 6
Author information 6
Abstract 6
Article content 8

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)



ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

Originality Statement

The author[s] declare that this article is their own work and to the best of their knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the published of any other published materials, except where due acknowledgement is made in the article. Any contribution made to the research by others, with whom author[s] have work, is explicitly acknowledged in the article.

Conflict of Interest Statement

The author[s] declare that this article was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright Statement

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

EDITORIAL TEAM

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

Article information

Check this article update (crossmark)

Check this article impact (*)

Save this article to Mendeley

(*) Time for indexing process is various, depends on indexing database platform

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

Classification of Rice Grains by Image Processing

Klasifikasi Butir Beras dengan Pemrosesan Citra

Zainab Naser Azeez, zainab.naser@iku.edu.iq, (1)
Computer Techniques Engineering Department, Imam AlKadhum University college, Nasiriyah, Thi-Qar, Iraq, Iraq

Aseel.A. Qasim, aseel.a.qasim@utq.edu.iq, (0)
Department of Information Technology, Faculty of Computer Science and Mathematics, University of Thi-Qar, Nasiriyah, Thi-Qar, Iraq, Iraq

Najlaa Muhammed Mohie, najlaamuhammed.eps@utq.edu.iq, (0)
Department of Computer Scienc, Faculty of Education for Pure Sciences, University of Thi-Qar, Nasiriyah,Thi-Qar, Iraq, Iraq

(1) Corresponding author

Abstract

General Background: The assessment of the surface quality of pre-treated rice grains is crucial for determining their market acceptability, storage stability, processing quality, and overall consumer satisfaction. Traditional evaluation methods are often time-consuming and yield subjective classifications. **Specific Background:** The lack of a comprehensive tagged image dataset hinders the application of advanced convolutional neural networks (CNN) for detailed damage classification of healthy rice grains. **Knowledge Gap:** Existing datasets and methods limit the effective exploration of sophisticated CNN models for categorizing rice types, particularly in identifying subtle damage characteristics. **Aims:** The study aims to create a robust rice grain classification system using image processing techniques, primarily deep learning algorithms, to improve the classification of rice varieties. **Results:** Utilizing a dataset of 75,000 images across five widely cultivated rice varieties in Turkey, we achieved classification accuracies of 100% for CNN, 99.95% for deep neural networks (DNN), and 99.87% for artificial neural networks (ANN). **Novelty:** The proposed approach represents a significant advancement in rice classification technology, employing a combination of image acquisition, feature extraction, and machine learning to streamline the process, effectively addressing the challenges faced in traditional methods. **Implications:** The findings underscore the potential for improved sorting and grading efficiency in the rice industry, facilitating better market outcomes and consumer satisfaction through enhanced quality control

Highlights:

1. Innovative: Uses CNN for accurate rice variety classification.
2. Data-Driven: Analyzes 75,000 images for enhanced quality evaluation.
3. Impactful: Increases efficiency in sorting and grading processes.

Keywords: Rice Classification, Image Processing, CNN, Machine Learning, Agricultural

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

Technology

Published date: 2024-10-15 00:00:00

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

Introduction

This work focuses on the development of image processing algorithms for the segmentation and identification ofthe rice grains. The utilization of image processing algorithms proves to be an efficient approach for measuring grain quality based on their size. This research proposes a method for categorizing and assessing rice grains by analyzing their size and form through the application of image processing methods [1]. More precisely, Use edge detection technology to identify the circumference of each particle [2] . This approach involves identifying the terminal points of each grain and thereafter use calipers to accurately Measure the rice's length and width.This approach takes a small amount of effort and incurs modest expenses. The conventional techniques employed for measuring grain form and size comprise the graphical method, dial micrometer, and grain shape test; nevertheless, these methods are time-consuming. This apparatus allows for the measurement of the width and length of a single grain at a time. Moreover, these procedures provide outcomes that are protracted, costly, and prone to human fallibility, necessitating a high level of precision to validate client requirements and surmount constraints in evidence [3]. Several studies have examined the morphological characteristics of grains, specifically focusing on their area and shape of the nucleus. Completed Nevertheless, the wide array of species exhibit such varied forms and sizes that it is impossible to establish a universal formula[4]. Categorization of all rice cultivars. Grain pictures are analyzed in this work to extract Fourier features, in addition to Spatial information, in order to achieve enhanced classification accuracy[5].

Utilizing picture processing strategies to discover rice grains can provide a treasured and green way of automating the classification of numerous rice types. Below are a sequence of pragmatic measures to hire image processing for the cause of rice grain identity: Gathering photographs of diverse rice kernels: In order to generate a dataset for the picture processing approach, it's far vital to accumulate a sizeable amount of photographs depicting diverse rice grains. Rice grains may be photographed using a digicam or smartphone, or as a substitute, snap shots may be accumulated through web platforms. Pre-processing of photos: Prior to doing image evaluation, it can be essential to preprocess the pics through removing any background noise or making changes to the lighting situations[6][7]. These duties encompass photograph cropping, resizing, and the use of filters to beautify assessment and coloration. Features Extraction method: After the pre-processing of the images, feature extraction techniques can be hired to discover the distinct attributes of every person grain of rice. For instance, area detection may be hired to check the morphology of a grain of rice, while coloration evaluation may be applied to assess the hue and saturation of rice.. Training the device mastering model: After the functions are acquired, the algorithm version can be trained to recognize distinct styles of rice grains through analyzing the unique elements within the photograph[8][9]. Analyze and improve the version: After training, you can check take a look at the version's accuracy the use of a validation dataset. It might be required to add greater information to the dataset or regulate the parameters so one can enhance the version's overall performance.

Overall, employing picture processing to determine several sorts of rice grains is a amazing and efficient technique for automating the system[10]. An accurate and reliable photograph-based totally rice grain identity machine may be advanced by way of imposing these practical processes.

Methods

The Structure of Rice Variety Recognition

Below is a schematic of the suggested rice grain identification system.

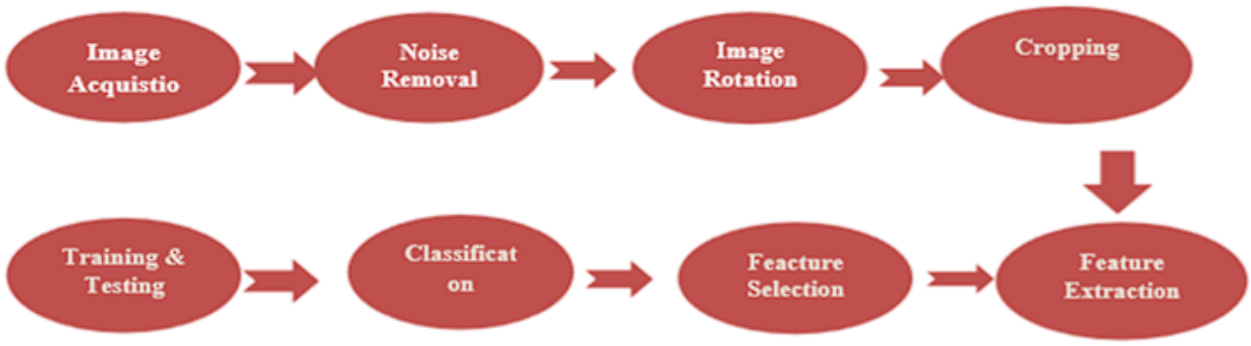


Figure 1. *schematic of the suggested rice grain identification system*

Image Acquisition

Rice is one of the most important cereals and has high levels of genetic diversity on a global scale. The two kinds of

these other Socks are separated from each other due to their certain qualities. Generally, These characteristics include of color, form, and texture. These distinctive qualities of different rice types also aid in the assessment and characterization of seed quality. The study utilized five types of rice, the most widely farmed varieties in Turkey. These types of rice are compiled in Table 1. At the moment, he possesses a collection of 75,000 photos, 15,000 of which are from each grain, according to his website. A second dataset with 106 features was also employed by us. These images provide 12 morphology, 4 form, and 90 color features for this informatics task. The feature dataset was used to create two distinct models using the (DNN) and (ANN) techniquesOn the basis of the picture dataset, a different model was created using the (CNN) technique. proceeded with the classification processes. Using the models' confusion matrix values, the rates of False Positives (FP) and False Negatives (FN) can be performed. Each model's results were displayed in tables. In the classification job, the models' accuracies were "100% for CNN, 99.95% for DNN, and 99.87% "for ANN. The study's conflicting models, which were used to categorize the rice varieties, can be efficiently applied in this domain, according to the results.

Imports:

from sklearn.mod\el_selection import train_test_split

import matplotlib from tensorflow

import keras.import matplotlib as plt, pyplot.sns

import PIL as img import seaborn image.

import pathli import cv2 import os

import numpy as np

import image as Image

Preparing Data:

linkcode

data_dir = "../input/rice-image-dataset/Rice_Image_Dataset" # Dataset s path

data_dir = pathlib.Path(data_dir)

data_dir

*-data_dir

The path object provides a number of ways to work with directories and files, such as listing the contents of a directory, checking for a file, and creating new directories. In this example, the 'data_dir' variable is used to find the dataset listing, that could then be used to load and pre-process the photos inside the dataset.

Arborio = list(data_dir.Glob('Arborio/*'))[:600]

basmati = listing(data_dir.Glob('Basmati/*'))[:600]

ipsala = list(data_dir.Glob('Ipsala/*'))[:600]

jasmine = list(data_dir.Glob('Jasmine/*'))[:600]

karacadag = list(data_dir.Glob('Karacadag/*'))[:600]

This code hundreds pics from a listing containing subdirectories of various forms of rice. The code defines 5 lists, one for every sort of rice, and makes use of the route object's glob() approach to locate all files in each subdirectory that in shape a specific pattern

For instance, trying to find 'data_dir.Glob('Arborio /*')` finds all documents inside the 'Arborio' subdirectory of the dataset listing. '*' is a wildcard that matches any filename. The ensuing list of files is then truncated Using '[:600]` to specify the first six hundred files. This limits the variety of snap shots uploaded from each subdirectory to 600.

The same procedure is repeated for the opposite 4 subdirectories ("basmati", "ipsala", "jasmine" and "kraracadag"). The resulting lists (" arborio ", " basmati", " ipsala", "jasmine", and

karacadag") comprise the file paths for the selected photographs from every subdirectory. ."

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

These file direction lists can then be used to load and pre-procedure pictures the use of a library consisting of

TensorFlow or OpenCV.

fig, ax = plt.subplots(ncols=5, figsize=(20,5))

fig.suptitle('Rice Categorarborio_image = img.imread(arborio[0])

basmati_image = img.imread(basmati[0])

ipsala_image = img.imread(ipsala[0])

jasmine_image = img.imread(jasmine[0])

karacadag_image = img.imread(karacadag[0])

ax[0].set_title('arborio')

ax[1].set_title('basmati')

ax[2].set_title('ipsala')

ax[4].set_title('karacadag')

ax[3].set_title('jasmine')

ax[0].imshow(arborio_image)

ax[1].imshow(basmati_image)

ax[2].imshow(ipsala_image)

ax[3].imshow(jasmine_image)

ax[4].imshow(karacadag_image)

Five subdiagrams, or columns, and a row of images are produced with the aid of this code; every column and photo represents a wonderful form of rice. The `subplots()` function of the `matplotlib.Pyplot module is used to produce the `fig` and `ax variables. The `ncols` parameter is ready to five to supply 5 columns.

The `suptitle()` method is then used to set the title of the figure to the Rice Class.

The following 5 lines of code load the first image of each of the 5 types of rice into separate

variables (`arborio_image `, `basmati_image` , `ipsala_image `, `jasmine_image` and

. "`karacadag_image`) using the `imread()` function of the module "matplotlib Image

The last five lines of code specify the title of each subplot using the `set_title()` method of

each axis object (ax`) and display each image in its corresponding subplot using the 'imshow()' method of each axis object. The resulting figure shows the first image of each type of rice in separate subplots .

df_images={

arborio' : arborio, '

'basmati' : basmati,

'ipsala' : ipsala,

'jasmine' : jasmine,

'karacadag': karacadag

This Code Defines Two Python `Df_Images` And `Df_Labels. These Dictionaries Are Used To Store The File Paths Of The Images And Their Corresponding Labels For The Different Types Of Rice.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

The `Df_Images` Dictionary Has Five Key-Value Pairs, Where Each Key Represents A Type Of Rice ('Arborio', 'Basmati', 'Ipsala', 'Jasmine', 'Karacadag') And Each Value Is A List Of File Paths To The Corresponding Rice Images For That Type. The File Paths In These Lists Were Previously Obtained

Using The `Glob()` Method From The `Pathlib` Module.

The `Df_Labels` Dictionary Also Has Five Key-Value Pairs, Where Each Key Represents A Type Of

Rice And Each Value Is An Integer Label Assigned To That Rice Type. The Labels Are Assigned In

Alphabetical Order, With 'Arborio' Assigned The Label 0, 'Basmati' Assigned The Label 1, 'Ipsala'

Assigned The Label 2, 'Jasmine' Assigned The Label 3, And 'Karacadag' Assigned The Label 4.

These Dictionaries Can Be Used To Create an image dataset with labels, where each image has a corresponding label attached to it. This can help train machine learning models to categorize various rice varieties based on their images..

Img = Cv2.Imread(Str(Df_Images['Arborio'][0]))

Img.Shape

This Code Reads The Image File Specified By The File Path At The First Index Of The List Of File

Paths For The 'Arborio' Rice Type In The `Df_Images` Dictionary. The File Path Is Converted To A

String Using The `Str()` Method.The `Imread()` Function From The Opencv (Cv2) Module Is Used To Read The Image From The File And Store It In The `Img` Variable.

The image's dimensions are represented by a tuple that is returned by the {shape} attribute of the `img} variable., In The Format (Height, Width, Channels). If The Image Is Gray Scale, The Third Element (Channels) Will Be 1, And If It Is Color, The Third Element Will Be 3 (For Rgb) Or 4 (For Rgba).

The `Img.Shape` Expression Returns This Tuple Of Dimensions.

X, Y= [ ], [ ]

For Label, Images In Df_Images.Items:

Two empty Python lists, {X} and {y}, are created by this code to hold the picture data and labels, respectively. The `items()` method is then used to traverse through each key-value pair in the `df_images } dictionary. It assigns the value (a list of file paths) to the variable images and the key to the variable `label} for each key-value pair.

Next, a `for` loop is used to cycle thru every file path in the `photos` listing. It uses the `imread()` function from the OpenCV (cv2) module to study the image for every file direction, storing it within the `img variable. The image is then resized to a brand new size of (224, 224), that is a common length the use of the `resize()` technique from the cv2 module. Is a common input size for many system gaining knowledge of fashions. The resized picture is stored inside the `resized_img' variable.

Finally, it appends the resized picture to the `X` listing and the corresponding label (obtained from the `df_labels` dictionary) to the `y` list. This technique is repeated for all photos in all rice categories, resulting in a list of resized photographs in `X` and their corresponding labels in `y'.

X = np.array(X).

X = X/255; y = np.array(y).

This code methods the facts accumulated in the preceding code block. The collection of resized photographs stored in X is turned to a NumPy array the use of the 'np.Array' command. ()` method from the NumPy module. This is needed seeing that NumPy arrays are the favored layout for data in many system learning frameworks.After that, the picture pixel values are normalized by means of dividing them by way of 255, which sets their scale to zero to one. To accomplish this, use the equation X/255. Machine mastering models can perform higher and converge more speedy while the pixel values are normalized. Lastly, `np.Array() is used to convert the labels in y to an aNumPy array so they can be utilized with device studying.Algorithms that require NumPy arrays as inputs.#Spliting into test val teach

This code block divides the data into three sets: training, validation, and testing.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

The data is split into three sets twice using the 'train_test_split' method from the Scikit-learn package. On the initial call to 'train_test_split}, the original data is divided into two sets: training and testing/validation. The second call separates the test sets and validation sets.

. The training and test/validation sets are created from the data arrays {X} and {y}, with the test/validation set having 25% of the original data. contain the split data that was produced. After that, the test and validation set is divided once more into distinct test and validation sets, with the validation set being 50% of the test and validation set. The split data that results is kept in the variables

Importing tensorflow.keras.applications and building a model MobileNet is loaded consecutively from Tensorflow.Keras.Models imported from Tensorflow.Keras.layers Small, Level

To create a neural network, the code imports the necessary modules from the TensorFlow Keras API.

architecture of networks. "MobileNet" is a (CNN) architecture that has already undergone training for picture categorization. The `tensorflow.keras.applications` module contains a number of pre-trained models that can be used as the foundation for a range of image-related activities. models that provide the framework for a variety of image-related tasks.

.. With the help of the class "sequential," we can build a model with a linear stack of layers.

Put differently, one layer's output is sent into the next layer as input.A fully connected layer is called "Dense." Every neuron in one layer is connected to every other layer's neuron by means of this kind of layer.

'Flatten` is a layer that converts from import tensorflow.keras.applications Importing MobileNet from tensorflow.keras.models import sequentially from tensorflow.keras.layers Compact, Level

the output of the previous layer's multi-dimensional tensor into a one-dimensional tensor that can be sent into a dense layer. These modules are frequently used to construct neural networks for tasks involving the classification of images. One potential application of the `MobileNet} architecture is as a feature extractor for a novel classification problem. By covering the previously trained model with additional layers, we can fine-tune the model to classify images according to our specific needs.

base_model = MobileNet(weights="imagenet", include_top=False, input_shape=(224, 224, 3).

The code creates a foundation model for image classification by importing the pre-trained 'MobileNet' CNN architecture from the 'tensorflow.keras.applications' module. The MobileNet model is pre-trained on the ImageNet dataset.

, which consists of more than a million labeled images in various categories. Given that "imagenet" is selected in the `weights` option, the pre-trained weights will be downloaded and applied to the model.

The `include_top` option is set to {False}, thus the final fully connected layer of the pre-trained model—which deals with classification—is not included. Rather, the base model{(None, 7, 7, 1024)} generates the 3D tensor with shape.

The form of the input photos that will be fed into the model is specified by the `input_shape} parameter. It is set to {(224, 224, 3)} in this instance, indicating that the model anticipates input images to have three color channels (RGB) and a width and height of 224 pixels.

summary() for base_model

A tabular assessment of the model architecture is provided by way of the summary()` feature, a way of the MobileNet model object. An review of the model's layers, each layer's output form, the range of trainable and non-trainable parameters, and the overall range of parameters are published. The pre-skilled `MobileNet` version, loaded with pre-skilled weights from the ImageNet dataset, is summarized on this code by means of `base_model.Precis()`. The model's layers are defined in depth inside the precis, along with every layer's output shape and the quantity of trainable and non-trainable layers.

Top_model = Sequential ()

top_model.add(Flatten(input_shape=base_model.output_shape[1:]))

top_model.add(Dense(32, activation="relu").

top_model.add(Dense(10, activation="softmax").

top_model.summary()

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

This code generates a new neural network version named 'top_model', so that you can be layered on top of the previously educated 'MobileNet' version. The 'Sequential()' technique creates an empty model item to which we can also upload layers. The 'top_model' begins with a 'Flatten()' layer, which converts the output of the 'base_model' into a single 1D vector. This layer is needed since the output of the 'base_model' is a 3-d tensor, however the completely linked layers we upload to the 'top_model' need a 1D enter. Following the 'Flatten()' layer, the 'top_model' constructs a fully linked 'Dense()' layer with 32 neurons and a'relu' activation feature. This layer learns a nonlinear function that converts the flattenrd enter into a brand new feature space, consisting of the output form of every layer, the wide variety of trainable and non-trainable parameters,and the whole range of parameters within the version.

transfer_model = Sequential()

transfer_model.add(base_model)

transfer_model.add(top_model)

transfer_model.layers[0].Trainable = False.

transfer_model.summary()

transfer_model.compile(optimizer="adam", loss=tf.keras.losses).SparseCategoricalCrossentropy(from_logits=True; metrics=['acc'])

History = transfer_model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val)).

This block of code defines a new model `transfer_model` using the pre-educated MobileNet version because the base and including a brand new pinnacle version on pinnacle it.

First, the pre-educated MobileNet model's 'base_model' is loaded. The version should be launched with pre-skilled weights from the ImageNet dataset, as indicated by means of putting the 'weights' option to "imagenet". Setting the 'include_top` argument to 'False` removes the model's top layer, which categorizes images into a thousand instructions in the ImageNet dataset. With the input_shape choice set to (224, 224, three), the model expects to get hold of enter pix with three RGB channels and 224 by means of 224 dimensions.

The definition of a new Sequential model, {top_model}, follows. The {Flatten} layer, the first layer of {top_model}, flattens the underlying model's output into a one-dimensional vector. Two 'Dense' levels with 32 and 10 come after this.

VISUALIZATION OF THE MODEL

sklearn. import metrics accuracy_score y_pred = transfer_model. # Predict & Print predictions from sklearn_predictions = model.

X_test, batch_size=64, verbose=1) # Converting Binary Classification array to 1's and 0's for Thresholding y_pred_bool = np. argmax(y_pred, axis=1)

accuracy_score(y_test, y_pred_bool) # Steps performed by the code.

1.It brings in the accuracy_score function from the sklearn. metrics module.

2. It takes the transfer_model predict() method which predicts using test set- X_test.

3. The numpy module's argmax() method is used to find the index of the biggest value in each prediction; in this case, the maximum value corresponds to the predicted class label. 4. The accuracy of the model is determined by comparing the expected labels ({y_pred_bool}) to the actual labels ({y_test}) using the 'accuracy_score()' function. The function returns the accuracy score as a float number.

Remember that the batch_size argument of the predict() function determines the number of samples that will be analyzed concurrently. this instance, 64 is the value set.

Result and Discussion

Preparing Data

PosixPath('../input/rice-image-dataset/Rice_Image_Dataset')

In this step of the code, it gives the names of each type of image entered to the rice, where it sorts the images.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

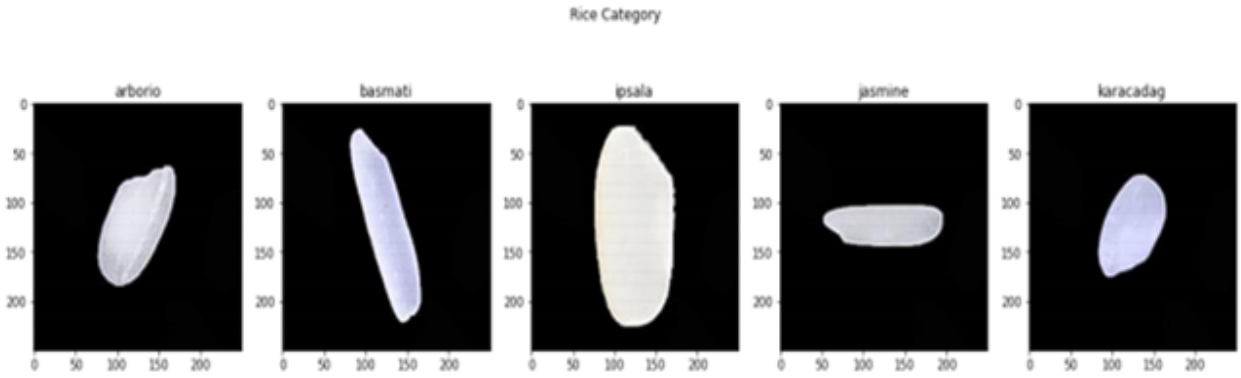


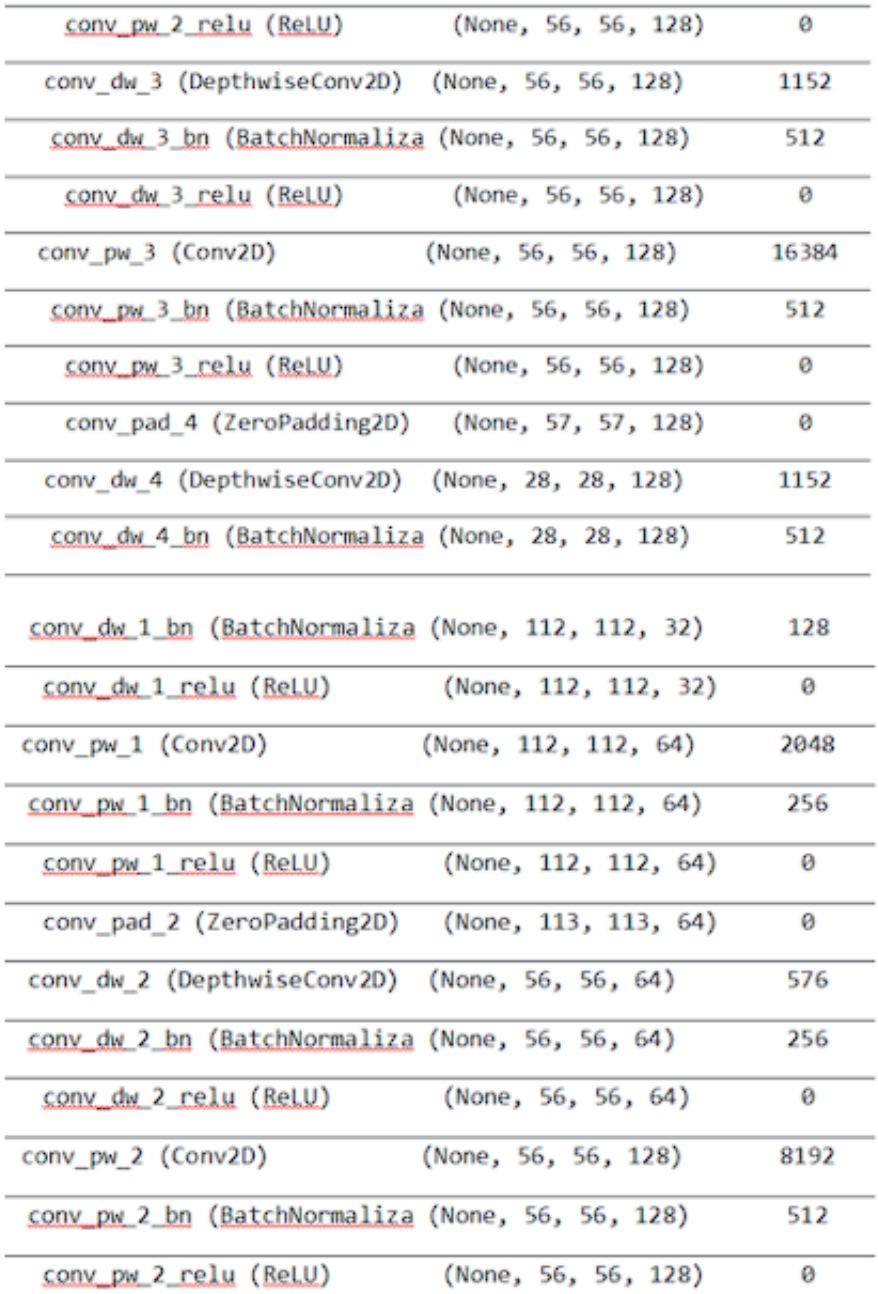
Figure 2. *rice Category*

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 64)	576
conv_dw_2_bn (BatchNormaliza	(None, 56, 56, 64)	256
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8192
conv_pw_2_bn (BatchNormaliza	(None, 56, 56, 128)	512

Figure 3.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

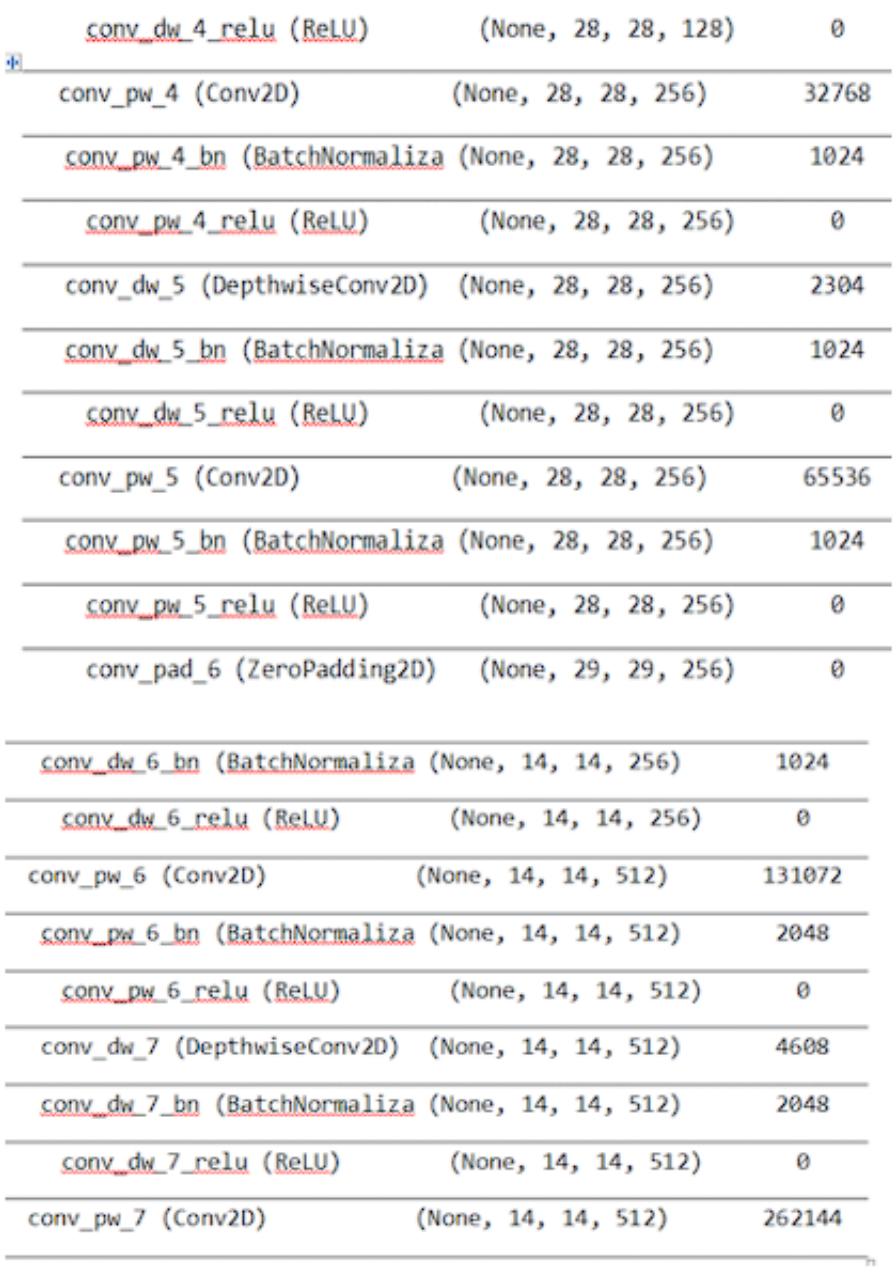


conv_pw_2_relu (ReLU)	(None, 56, 56, 128)	0
conv_dw_3 (DepthwiseConv2D)	(None, 56, 56, 128)	1152
conv_dw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_dw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pw_3 (Conv2D)	(None, 56, 56, 128)	16384
conv_pw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_pw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, 57, 57, 128)	0
conv_dw_4 (DepthwiseConv2D)	(None, 28, 28, 128)	1152
conv_dw_4_bn (BatchNormaliza	(None, 28, 28, 128)	512
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 64)	576
conv_dw_2_bn (BatchNormaliza	(None, 56, 56, 64)	256
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8192
conv_pw_2_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_pw_2_relu (ReLU)	(None, 56, 56, 128)	0

Figure 4.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)



conv_dw_4_relu (ReLU)	(None, 28, 28, 128)	0
conv_pw_4 (Conv2D)	(None, 28, 28, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_4_relu (ReLU)	(None, 28, 28, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, 28, 28, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_dw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pw_5 (Conv2D)	(None, 28, 28, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, 29, 29, 256)	0
conv_dw_6_bn (BatchNormaliza	(None, 14, 14, 256)	1024
conv_dw_6_relu (ReLU)	(None, 14, 14, 256)	0
conv_pw_6 (Conv2D)	(None, 14, 14, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_6_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_7 (Conv2D)	(None, 14, 14, 512)	262144

Figure 5.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

conv_dw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_dw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_11 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 15, 15, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, 7, 7, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, 7, 7, 512)	2048
conv_dw_12_relu (ReLU)	(None, 7, 7, 512)	0
conv_pw_12 (Conv2D)	(None, 7, 7, 1024)	524288
conv_pw_12_bn (BatchNormaliz	(None, 7, 7, 1024)	4096

Figure 6.

VISUALIZATION OF THE MODEL

	precision	recall	f1-score	support
0	0.98	0.99	0.99	117
1	1.00	0.98	0.99	123
2	1.00	1.00	1.00	106
3	0.98	0.99	0.99	112
4	0.99	0.99	0.99	104
accuracy			0.99	562
macro avg	0.99	0.99	0.99	562
weighted avg	0.99	0.99	0.99	562

Figure 7.

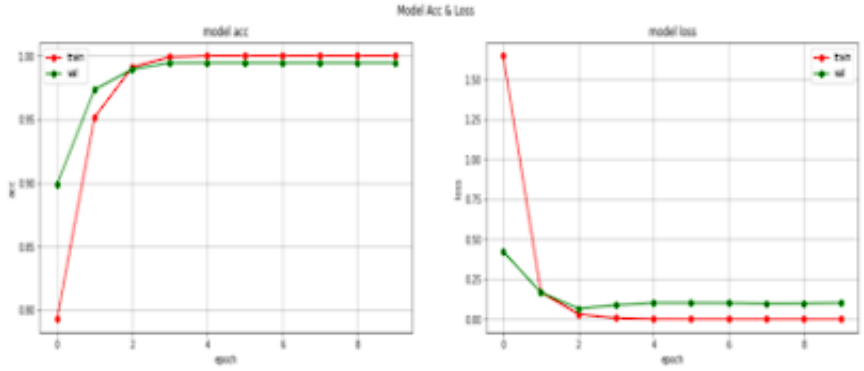


Figure 8.

Academia Open
Vol 9 No 2 (2024): December
DOI: 10.21070/acopen.9.2024.10303 . Article type: (Food Science)

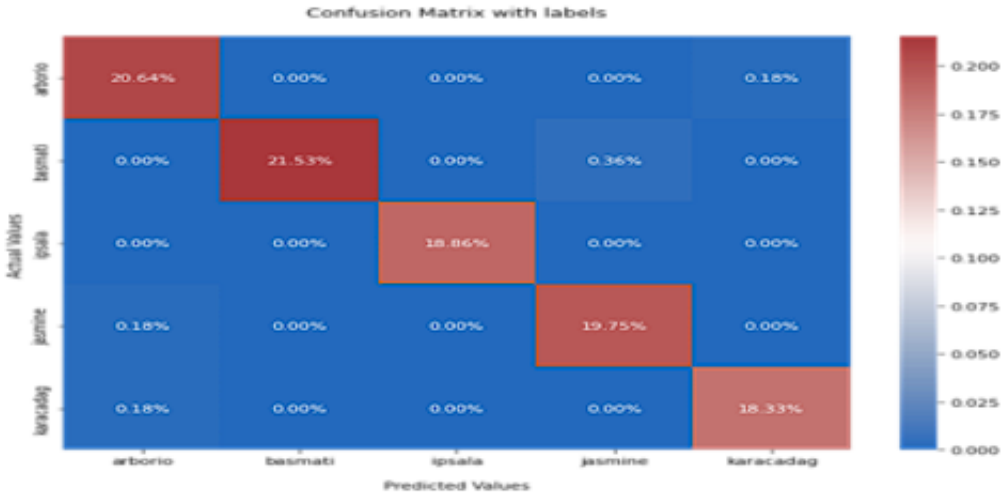


Figure 9.

Conclusion

In the end, using Python image processing to automate the tedious and time-consuming operation of sorting and grading rice grains can be quite helpful. Computer vision and machine learning techniques can be utilized for rice grain feature recognition based on forms, sizes, colors, and textures.

Each Image Can Be Used As Labeling An Element Of A Cell In These Dictionaries, So That Allowinary Images In Source To Create Dataset Of Labeled Images.

Labelled With Related Label. This can prove beneficial for training ML models to identify the species of rice using it's image.Img = Cv2. imread(str(Df_Images['Arborio'][0]))ImgFormMany image processing methods, such as edge detection, morphological operations, and segmentation may be used to obtain this data. We use computer learning techniques like CNN, SVM, and KNN to categorize rice grains based on their properties. Python supports OpenCV, Scikit-image, TensorFlow, and other image processing and machine learning libraries and tools. We can easily develop the image processing and machine learning pipelines for rice grain categorization by using these libraries.

Overall, classifying rice grains by image processing in Python is a promising area of research that can have significant implications for the rice industry, by improving the efficiency and accuracy of rice grain sorting and grading.

References

1. . A. Pasoriya, S. Chavhan, S. Gotarkar, and R. C. Dharmik, "Rice Quality Analysis Using Image," in Rice Quality Analysis Using Image, R. C. Dharmik, S. Chavhan, S. Gotarkar, and A. Pasoriya, Eds. New Delhi, India: National Institute of Technology, 2022, pp. 158-164.
2. . C. C. Hortineal, J. R. Balbin, J. C. Fausto, A. D. Catli, K. J. R. Cui, J. A. F. Tan, and E. O. S. Zunega, "Milled Rice Grain Grading Using Raspberry Pi With Image Processing and Support Vector Machines With Adaptive Boosting," in Proc. IEEE, 2020.
3. . H. O. Velesaca, P. L. Suárez, R. Mira, and A. D. Sappa, "Computers and Electronics in Agriculture," Computers and Electronics in Agriculture, vol. 177, 2021.
4. . I. Chatnuntawech, K. Tantisantisom, P. Khanchaitit, T. Boonkoom, B. Bilgic, and E. Chuangsuwanich, "Rice Classification Using Spatio-Spectral Deep Convolutional Neural Network," IJERT, vol. 9, no. 4, 2020.
5. . J. S. Aulakh and Dr. V. K. Banga, "Quality Assessment of Rice," Indian Journal of Agricultural Research, vol. 47, no. 2, pp. 116-120, 2012.
6. . P. Patil, "Reliable Quality Analysis of Indian Basmati," IJERT, vol. 3, no. 6, 2014.
7. . S. K. Dhyawa, A. Srivastava, S. Pal, and P. L. Bhutia, "Identification and Classification of Rice Varieties Using Image Processing," International Journal of Advanced Research in Computer Engineering & Technology, vol. 2, no. 5, 2014.
8. . W. P. N. W. M. Tahir, N. Hussin, Z. Z. Htike, and W. Y. N. Naing, "Rice Grading Using Image Processing," ARPN Journal of Engineering and Applied Sciences, vol. 10, no. 20, pp. 9634-9638, 2015.
9. . Z. Z. Htike and W. Y. N. Naing, W. P. N. W. M. Tahir, and N. Hussin, "Rice Grading Using Image Processing," ARPN Journal of Engineering and Applied Sciences, vol. 10, no. 20, pp. 9639-9643, 2015.

10. . C. V. Maheshwari, K. R. Jain, and C. K. Modi, "Non-Destructive Quality Analysis of Indian Gujarat-17 Oryza Sativa SSP Indica (Rice) Using Image Processing," International Journal of Computational Engineering Science, vol. 2, pp. 48-54, 2012.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo