

**LAPORAN FINAL PROJECT**  
**IDENTIFIKASI CITRA DAUN MENGGUNAKAN CNN**



**Mata Kuliah : Pengolahan Citra Digital**

**Anggota Kelompok**

Ayu Widya Agata	19081010005
Azka Avicenna Rasjid	20081010115
Farkhan	20081010060

**PROGRAM STUDI INFORMATIKA**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"**  
**JAWA TIMUR**  
**2022**

## **KATA PENGANTAR**

Puji dan syukur penulis panjatkan kepada Allah SWT karena atas karunianya penulis dapat menyelesaikan laporan final proyek dengan judul “Identifikasi Citra Daun Menggunakan CNN” ini sebagai hasil perkuliahan mata kuliah Pengolahan Citra Digital.

Laporan ini dapat diselesaikan tepat pada waktunya tidak terlepas dari bantuan dan dukungan dari berbagai pihak, yakni Eva Yulia Puspaningrum, S.Kom, M.Kom selaku dosen pengampu, dan teman-teman anggota kelompok. Untuk itu penulis ucapkan terima kasih atas kontribusi bantuan dalam berbagai bentuk.

Penulis menyadari bahwa masih banyak kesalahan dalam penyusunan laporan ini. Maka dari itu penulis sangat mengharapkan kritik dan saran seluas-luasnya dari pembaca yang kemudian akan penulis jadikan sebagai evaluasi. Demikian semoga laporan final proyek ini bisa diterima sebagai ide atau gagasan yang menambah kekayaan intelektual dan dapat bermanfaat bagi pembaca dan juga untuk penulis sendiri.

Surabaya, 19 Desember 2022

Penulis

## DAFTAR ISI

KATA PENGANTAR .....	ii
DAFTAR ISI.....	iii
BAB I.....	1
A. Deskripsi Program .....	1
B. Dataset.....	1
C. Tujuan .....	1
BAB II.....	2
A. Metodologi Penelitian.....	2
B. Explore Data Analysis .....	3
C. Deep Learning (Implementasi CNN).....	9
D. Hasil Uji Program .....	20
BAB III .....	23
DAFTAR PUSTAKA .....	24

# BAB I

## PENDAHULUAN

### A. Deskripsi Program

Sistem identifikasi citra daun merupakan sistem yang pengaplikasiannya hampir mirip dengan sistem pengenalan wajah, sistem pengidentifikasian citra daun ini sendiri terdiri dari tahap deteksi dan klasifikasi. Program identifikasi citra daun ini dibuat menggunakan bahasa pemrograman *python* dengan *library* seperti *matplotlib*, *tensorflow* dan *keras* (Kuriniawan, 2019).

Di dalam penelitian ini, kelompok kami menggunakan metode CNN yang digunakan untuk mengklasifikasikan objek citra daun pepaya, daun singkong, dan daun ganja. CNN adalah sebuah metode untuk memproses data dalam bentuk beberapa array, contohnya yaitu gambar berwarna yang terdiri dari tiga array 2D yang mengandung intensitas piksel dalam tiga jenis warna. *Convolutional Neural Networks* (ConvNets) merupakan penerapan dari *Artificial Neural Networks* (ANN) yang lebih istimewa dan saat ini diklaim sebagai model terbaik untuk memecahkan masalah pengenalan objek. Secara teknis, *convolutional network* memiliki arsitektur yang dapat dilatih dan terdiri dari beberapa tahap.

Program ini difokuskan melakukan identifikasi tiga kelas atau kategori daun dikarenakan semua bentuk daun ini hampir memiliki kemiripan, yang mana kita akan mengolah data untuk dapat membedakan daun-daun tersebut dengan menggunakan metode *deep learning convolutional neural network* (CNN).

### B. Dataset

Program ini menggunakan 130 gambar sebagai *dataset* yang berupa 50 gambar untuk daun singkong, 50 gambar untuk daun ganja, dan 30 gambar untuk daun pepaya. Dataset tersebut didapatkan melalui *collect* secara mandiri dengan *background* gambar berwarna putih.

### C. Tujuan

Tujuan dibuatnya program identifikasi citra daun menggunakan algoritma *convolutional neural network* (CNN) ini, di antaranya:

1. Mengetahui cara kerja dan hasil akurasi algoritma CNN terhadap citra daun
2. Melakukan identifikasi atau klasifikasi citra daun berdasarkan tiga kelas klasifikasi, yaitu daun singkong, daun ganja dan daun pepaya.

## BAB II

### HASIL DAN PEMBAHASAN

Berikut metodologi penelitian dan penjelasan beberapa potongan kode dari program identifikasi citra daun menggunakan algoritma *convolutional neural network* (CNN) di antaranya:

#### A. Metodologi Penelitian



Gambar 1. Langkah-langkah pembuatan program

Dari gambar 1. menjelaskan alur proses pembuatan program identifikasi citra daun menggunakan CNN. Proses diawali dengan menentukan topik, mengidentifikasi masalah, melakukan studi pustaka, pengumpulan *dataset*, perancangan *preprocessing*, perancangan CNN, pengujian model, dan yang terakhir yaitu hasil dari uji coba program.

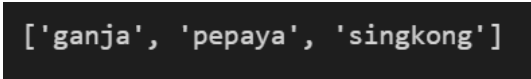
## B. Explore Data Analysis

Kode:

```
import warnings
import os
warnings.filterwarnings('ignore')
# get all the paths
data_dir_list = os.listdir('dataset-2')
print(data_dir_list)
path, dirs, files = next(os.walk("dataset-2"))
file_count = len(files)
# print(file_count)
```

Pada potongan kode diatas menjelaskan mengimport dataset berupa citra daun yang berada dalam folder “dataset-2” . Hasil output dari potongan kode diatas ada nama dari masing-masing folder yang berada dalam folder dataset.

Output:



```
['ganja', 'pepaya', 'singkong']
```

Gambar 2. Output hasil

Kode:

```
import warnings
# make new base directory
original_dataset_dir = 'dataset-2'
base_dir = 'leaves-data-2'
os.mkdir(base_dir)
```

Pada potongan kode diatas menjelaskan pembuatan folder baru dengan nama “leaves-data-2” yang digunakan untuk menyimpan data yang akan digunakan.

Kode:

```
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
```

Pada potongan kode diatas menjelaskan pembuatan dua folder dengan nama “train” dan “validation” yang ada di dalam folder “leaves-data-2” sebelumnya.

Kode:

```
# create two folders in base_dir train and validation

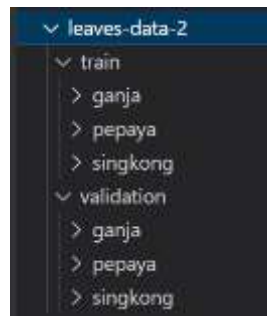
train_dir = os.path.join(base_dir, 'train')
os.mkdir(train_dir)
validation_dir = os.path.join(base_dir, 'validation')
os.mkdir(validation_dir)

# under train folder create 4 folder (ganja, singkong, pepaya)
train_ganja_dir = os.path.join(train_dir, 'ganja')
os.mkdir(train_ganja_dir)
train_singkong_dir = os.path.join(train_dir, 'singkong')
os.mkdir(train_singkong_dir)
train_pepaya_dir = os.path.join(train_dir, 'pepaya')
os.mkdir(train_pepaya_dir)

# under validation folder create 4 folder (ganja, singkong, pepaya)
validation_ganja_dir = os.path.join(validation_dir, 'ganja')
os.mkdir(validation_ganja_dir)
validation_singkong_dir = os.path.join(validation_dir, 'singkong')
os.mkdir(validation_singkong_dir)
validation_pepaya_dir = os.path.join(validation_dir, 'pepaya')
os.mkdir(validation_pepaya_dir)
```

Pada potongan kode diatas menjelaskan pembuatan empat folder pada masing-masing folder “train” dan “validation”. Folder ini nantinya digunakan untuk menyimpan tiga kelas daun yaitu daun ganja, daun singkong dan daun pepaya.

Output:



Gambar 3. Output hasil

Kode:

```
def split_data(SOURCE, TRAINING, VALIDATION, SPLIT_SIZE):
    files = []
    for filename in os.listdir(SOURCE):
        file = SOURCE + filename
        if os.path.getsize(file) > 0:
            files.append(filename)
        else:
            print(filename + " is zero length, so ignoring.")

    training_length = int(len(files) * SPLIT_SIZE)
    valid_length = int(len(files) - training_length)
    shuffled_set = random.sample(files, len(files))
    training_set = shuffled_set[0:training_length]
    valid_set = shuffled_set[:valid_length]

    for filename in training_set:
        this_file = SOURCE + filename
        destination = TRAINING + filename
        copyfile(this_file, destination)

    for filename in valid_set:
        this_file = SOURCE + filename
        destination = VALIDATION + filename
        copyfile(this_file, destination)
```

Pada potongan kode diatas menjelaskan melakukan split data pada folder data training dan juga data validation.



Kode:

```
GANJA_SOURCE_DIR = 'dataset-2/ganja/'
TRAINING_GANJA_DIR = 'leaves-data-2/train/ganja/'
VALID_GANJA_DIR = 'leaves-data-2/validation/ganja/'

SINGKONG_SOURCE_DIR = 'dataset-2/singkong/'
TRAINING_SINGKONG_DIR = 'leaves-data-2/train/singkong/'
VALID_SINGKONG_DIR = 'leaves-data-2/validation/singkong/'

PEPAYA_SOURCE_DIR = 'dataset-2/pepaya/'
TRAINING_PEPAYA_DIR = 'leaves-data-2/train/pepaya/'
VALID_PEPAYA_DIR = 'leaves-data-2/validation/pepaya/'
```

Pada potongan kode diatas menjelaskan penentuan path folder masing-masing kelas daun mulai dari source data, data yang akan di training dan data yang akan dilakukan validasi.

Kode:

```
import os
import random
from shutil import copyfile

split_size = 0.85

split_data(GANJA_SOURCE_DIR, TRAINING_GANJA_DIR, VALID_GANJA_DIR, split_size)
split_data(SINGKONG_SOURCE_DIR, TRAINING_SINGKONG_DIR, VALID_SINGKONG_DIR,
split_size)
split_data(PEPAYA_SOURCE_DIR, TRAINING_PEPAYA_DIR, VALID_PEPAYA_DIR, split_size)
```

Pada potongan kode diatas menjelaskan melakukan split data sebanyak 0.85 dari keseluruhan data (total data, training data, dan validation data).

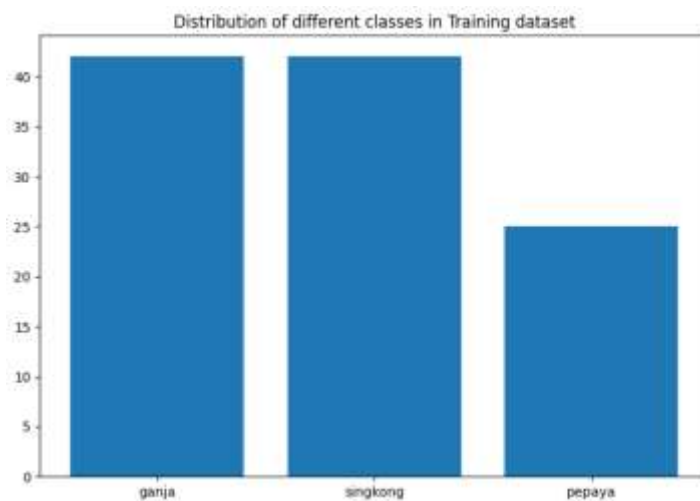
Kode:

```
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.image import imread
import pathlib

image_folder = ['ganja', 'singkong', 'pepaya']
nimgs = {}
for i in image_folder:
    nimages = len(os.listdir('leaves-data-2/train/' + i + '/'))
    nimgs[i] = nimages
plt.figure(figsize=(9, 6))
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
plt.xticks(range(len(nimgs)), list(nimgs.keys()))
plt.title('Distribution of different classes in Training dataset')
plt.show()
```

Pada potongan kode diatas menjelaskan cara menampilkan histogram dari masing-masing dataset pada folder “train”. Untuk menampilkan histogram, dilakukan *pengimport an library* terlebih dahulu kemudian mendefinisikan nama folder dan menampilkannya pada output.

Output:



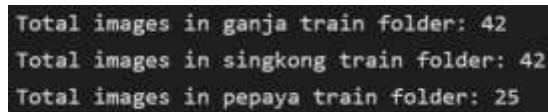
Gambar 4. Output hasil

Kode:

```
for i in image_folder:
    print('Total images in {} train folder: {}'.format(i, len(os.listdir('leaves-
data-2/train/' + i + '/'))))
```

Pada potongan kode diatas menjelaskan cara menghitung jumlah data yang ada pada masing-masing kelas daun dalam folder data train.

output:



```
Total images in ganja train folder: 42
Total images in singkong train folder: 42
Total images in pepaya train folder: 25
```

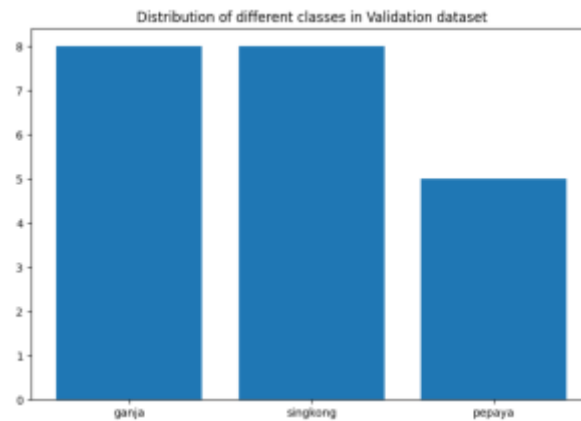
Gambar 5. Output hasil

Kode:

```
nimgs = {}
for i in image_folder:
    nimages = len(os.listdir('leaves-data-2/validation/' + i + '/'))
    nimgs[i] = nimages
plt.figure(figsize=(9, 6))
plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
plt.xticks(range(len(nimgs)), list(nimgs.keys()))
plt.title('Distribution of different classes in Validation dataset')
plt.show()
```

Pada potongan kode diatas menjelaskan cara menampilkan histogram dari masing-masing dataset pada folder “validation”.

output:



Gambar 6. Output hasil

Kode:

```
for i in image_folder:  
    print('Total images in {} validation folder: {}'.format(i,  
len(os.listdir('leaves-data-2/validation/' + i + '/'))))
```

Pada potongan kode diatas menjelaskan cara menghitung jumlah data yang ada pada masing-masing kelas daun dalam folder data validation.

output:

```
Total images in ganja validation folder: 8  
Total images in singkong validation folder: 8  
Total images in pepaya validation folder: 5
```

Gambar 7. Output hasil

### C. Deep Learning (Implementasi CNN)

Kode:

```
from tensorflow.keras.optimizers import Adam  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout  
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

Pada potongan kode diatas menjelaskan dilakukan pengimport an library tensorflow dan keras yang nantinya digunakan untuk implementasi algoritma convolutional neural network (CNN).

Kode:

```
img_width, img_height = 256, 256
batch_size = 16
```

Pada potongan kode diatas menjelaskan cara mensetting ukuran gambar yaitu dengan ukuran panjang 256 dan lebar 256.

Kode:

```
TRAINING_DIR = 'leaves-data-2/train/'

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=30,
                                   zoom_range=0.4,
                                   horizontal_flip=True)

train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    target_size=(img_width,
img_height))
```

Pada potongan kode diatas menjelaskan cara melakukan data augmentasi yang digunakan untuk menambahkan dataset yang sebelumnya. Setelah itu, ditampilkan output berupa total gambar dari semua kelas/kategori daun.

output:

```
Found 109 images belonging to 3 classes.
```

Gambar 8. Output hasil

Kode:

```
VALIDATION_DIR = 'leaves-data-2/validation/'

validation_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,

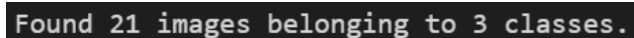
batch_size=batch_size,

class_mode='categorical',

target_size=(img_width, img_height))
```

Pada potongan kode diatas menjelaskan menjelaskan cara melakukan data augmentasi yang digunakan untuk menambahkan dataset yang sebelumnya. Setelah itu, ditampilkan output berupa total gambar dari semua kelas/kategori daun.

Output:

A screenshot of a terminal window with a dark background. The text 'Found 21 images belonging to 3 classes.' is displayed in a light-colored monospace font.

Gambar 9. Output hasil

Kode:

```
callbacks = EarlyStopping(monitor='val_loss', patience=5, verbose=1,
mode='auto')
# autosave best model
best_model_file = 'best_model.h5'
best_model = ModelCheckpoint(best_model_file, monitor='val_accuracy', verbose =
1, save_best_only = True)
```

Pada potongan kode diatas menjelaskan cara menyimpan semua data yang telah di train dan validation sebagai file h5 dengan nama file “best\_model.h5”.

Kode:

```
model = Sequential()
model.add(Conv2D(16, (3, 3), activation='relu', input_shape=(img_width,
img_height, 3)))
model.add(MaxPooling2D(2, 2))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()
```

Pada potongan kode diatas menjelaskan pengimplementasian algoritma convolutional neural network menggunakan convolusi 2D.

Output:

Output exceeds the [size limit](#), open the full output data in a [text editor](#).

Layer (type)	Output Shape	Param #
conv2d_0 (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d_0 (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_1 (Conv2D)	(None, 126, 126, 32)	8640
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_2 (Conv2D)	(None, 60, 60, 64)	18064
conv2d_3 (Conv2D)	(None, 58, 58, 64)	16928
max_pooling2d_2 (MaxPooling2D)	(None, 29, 29, 64)	0
conv2d_4 (Conv2D)	(None, 27, 27, 128)	72056
conv2d_5 (Conv2D)	(None, 25, 25, 128)	147088
...		
Total params: 1,166,400		
Trainable params: 1,166,400		
Non-trainable params: 0		

Gambar 10. Output hasil

Kode:

```
model.compile(optimizer='Adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

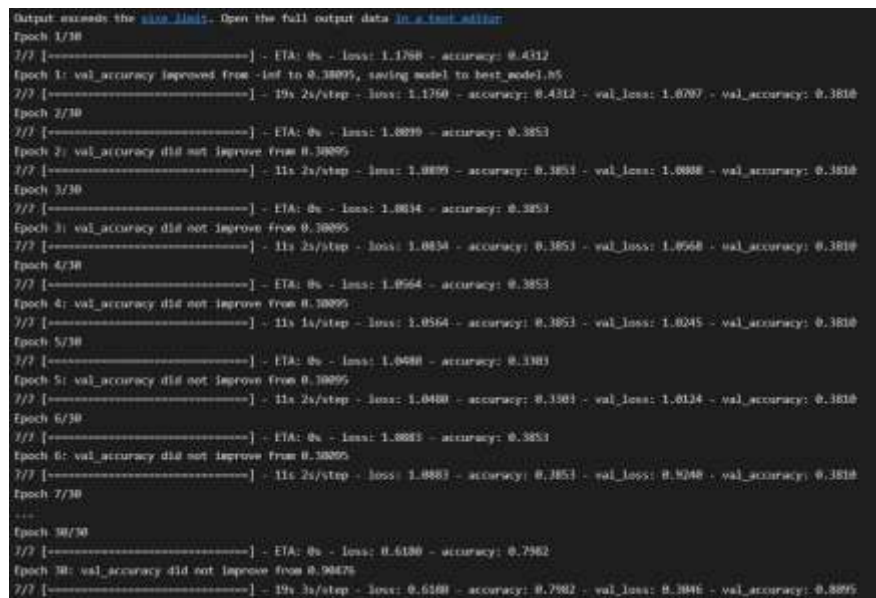
Pada potongan kode diatas menjelaskan pengoptimalan menggunakan adam.

Kode:

```
history = model.fit_generator(train_generator,
                             epochs=30,
                             verbose=1,
                             validation_data=validation_generator,
                             callbacks=[best_model])
```

Pada potongan kode diatas menjelaskan cara menentukan epoch yaitu berapa kali algoritma deep learning bekerja melewati seluruh dataset baik secara forward maupun backward.

Output:



```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/30
1/1 [-----] - ETA: 0s - loss: 1.1760 - accuracy: 0.4312
Epoch 1: val_accuracy improved from -inf to 0.38095, saving model to best_model.h5
1/1 [-----] - 19s 2s/step - loss: 1.1760 - accuracy: 0.4312 - val_loss: 1.6707 - val_accuracy: 0.3810
Epoch 2/30
1/1 [-----] - ETA: 0s - loss: 1.0890 - accuracy: 0.3853
Epoch 2: val_accuracy did not improve from 0.38095
1/1 [-----] - 11s 2s/step - loss: 1.0890 - accuracy: 0.3853 - val_loss: 1.6888 - val_accuracy: 0.3810
Epoch 3/30
1/1 [-----] - ETA: 0s - loss: 1.0814 - accuracy: 0.3853
Epoch 3: val_accuracy did not improve from 0.38095
1/1 [-----] - 11s 2s/step - loss: 1.0834 - accuracy: 0.3853 - val_loss: 1.6968 - val_accuracy: 0.3810
Epoch 4/30
1/1 [-----] - ETA: 0s - loss: 1.0564 - accuracy: 0.3853
Epoch 4: val_accuracy did not improve from 0.38095
1/1 [-----] - 11s 1s/step - loss: 1.0564 - accuracy: 0.3853 - val_loss: 1.6245 - val_accuracy: 0.3810
Epoch 5/30
1/1 [-----] - ETA: 0s - loss: 1.0488 - accuracy: 0.3303
Epoch 5: val_accuracy did not improve from 0.38095
1/1 [-----] - 11s 2s/step - loss: 1.0488 - accuracy: 0.3303 - val_loss: 1.6124 - val_accuracy: 0.3810
Epoch 6/30
1/1 [-----] - ETA: 0s - loss: 1.0883 - accuracy: 0.3853
Epoch 6: val_accuracy did not improve from 0.38095
1/1 [-----] - 11s 2s/step - loss: 1.0883 - accuracy: 0.3853 - val_loss: 0.9248 - val_accuracy: 0.3810
Epoch 7/30
...
Epoch 30/30
1/1 [-----] - ETA: 0s - loss: 0.6180 - accuracy: 0.7982
Epoch 30: val_accuracy did not improve from 0.90476
1/1 [-----] - 19s 3s/step - loss: 0.6180 - accuracy: 0.7982 - val_loss: 0.3046 - val_accuracy: 0.8895
```

Gambar 11. Output hasil



Kode:

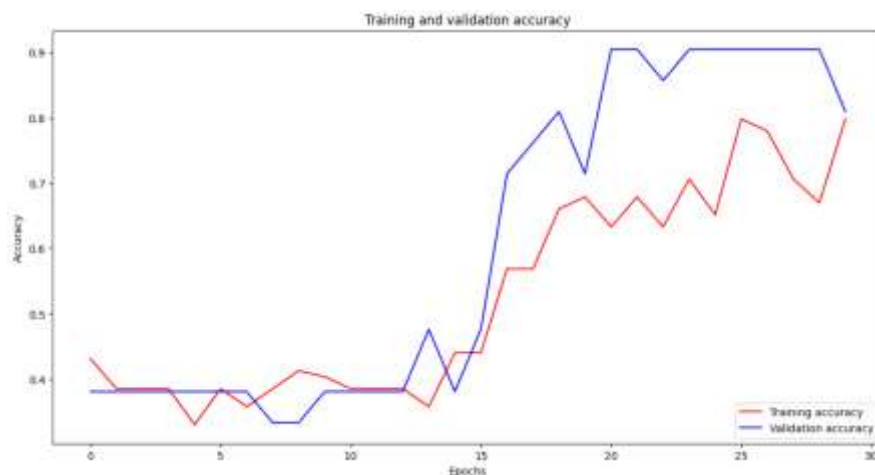
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

fig = plt.figure(figsize=(14, 7))
plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc='lower right')
plt.show()
```

Pada potongan kode diatas menjelaskan cara menampilkan diagram hasil akurasi dari data training dan data validation tiap kelas/kategori.

Output:



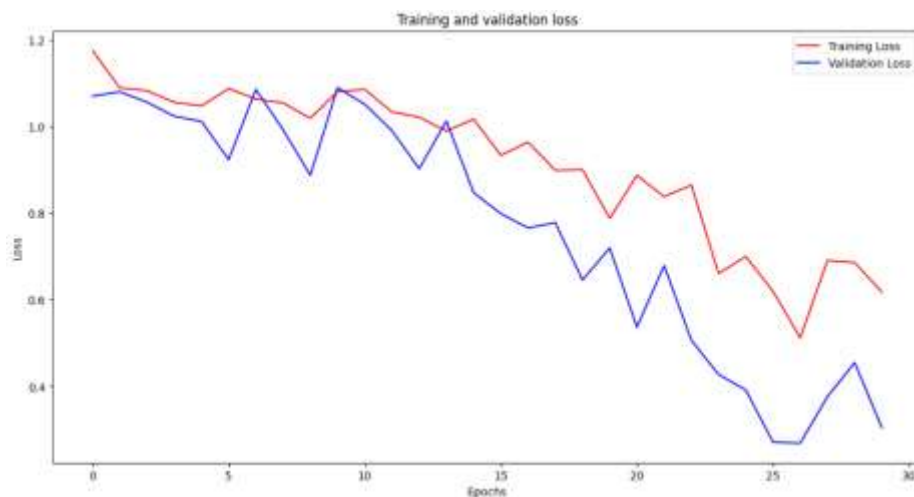
Gambar 12. Output hasil

Kode:

```
fig2 = plt.figure(figsize=(14, 7))
plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and validation loss')
plt.legend(loc='upper right')
plt.show()
```

Pada potongan kode diatas menjelaskan cara menampilkan diagram hasil loss dari data training dan data validation tiap kelas/kategori.

Output:



Gambar 13. Output hasil

Kode:

```
# show accuracy and loss
score = model.evaluate_generator(validation_generator)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Pada potongan kode diatas menjelaskan cara menampilkan nilai akurasi dan loss dari data yang telah diolah.

Output:

```
Test loss: 0.30458614230155945
Test accuracy: 0.8095238208770752
```

Gambar 14. Output hasil

Kode:

```
from keras import utils
from keras.models import load_model
import numpy as np
```

Pada potongan kode diatas menjelaskan pengimport an *library* keras dan numpy.

Kode:

```
# import libraries
from tkinter import *
from tkinter import filedialog
from PIL import ImageTk, Image

# create root window
root = Tk()
root.title("Deteksi Daun")
root.geometry("500x400")

# create fungsi enable button
def enable_button():
    loadCitra.config(state=NORMAL)
    grayscale.config(state=NORMAL)
    identifikasi.config(state=NORMAL)

# function load citra
def load_citra():
    root.filename =
filedialog.askopenfilename(initialdir='D:/UPN/Semester/semester5',
title="Select A File", filetypes=(("jpg files", "*.jpg"), ("all files", "*.*")))

# load image
```

```

ori_image = Image.open(root.filename)
# resize image
resized_img1 = ori_image.resize((150, 150), Image.ANTIALIAS)

empt1 = ImageTk.PhotoImage(resized_img1)
empty1 = Label(root, image=empt1)
empty1.image = empt1

# pasangkan gambar
empty1.grid(row=1, column=1, rowspan=3, padx=5)

# get just file name without path
file_name = root.filename.split("/")[-1]
# print file name in input
name.insert(0, file_name)

# function grayscale image
def grayscale_image():
    gray_img = Image.open(root.filename).convert('L')
    resized_img2 = gray_img.resize((150, 150), Image.ANTIALIAS)

    empt2 = ImageTk.PhotoImage(resized_img2)
    empty2 = Label(root, image=empt2)
    empty2.image = empt2

    # pasangkan gambar
    empty2.grid(row=1, column=2, rowspan=3, padx=5)

# function identifikasi
def identif():
    # load model
    model = load_model(best_model_file)

    # load image
    # img = Image.open(root.filename).convert('L')
    # resize image
    img = utils.load_img(root.filename, target_size=(256, 256))

```

```

x = utils.img_to_array(img)
x = np.expand_dims(x, axis=0)
images = np.vstack([x])
classes = model.predict(images, batch_size=10)
class_pred = np.argmax(classes, axis=1)

# get class name
if class_pred == 0:
    class_name = "Daun ganja"
elif class_pred == 1:
    class_name = "Daun pepaya"
else:
    class_name = "Daun singkong"

# print class name in input
result.insert(0, class_name)

# buat fungsi reset
# fungsi ini digunakan untuk menghapus semua hasil dari load citra
def reset():
    # hapus semua hasil dari load citra
    empty1 = Label(root, image=empt1)
    empty1.image = empt1

    empty2 = Label(root, image=empt2)
    empty2.image = empt2

    empty1.grid(row=1, column=1, rowspan=3, padx=5)
    empty2.grid(row=1, column=2, rowspan=3, padx=5)

    name.delete(0, END)
    result.delete(0, END)

# create menu button
loadNet = Button(root, text="Load Net", padx=10, pady=5, fg="white",
bg="#263D42", command=enable_button)
loadCitra = Button(root, text="Load Citra", padx=10, pady=5, fg="white",
bg="#263D42", state="disable", command=load_citra)

```

```

grayscale = Button(root, text="Grayscale", padx=10, pady=5, fg="white",
bg="#263D42", state="disable", command=grayscale_image)
identifikasi = Button(root, text="Identifikasi", padx=10, pady=5, fg="white",
bg="#263D42", state="disable", command=identif)
reset = Button(root, text="Reset", padx=10, pady=5, fg="white", bg="#263D42",
command=reset)

name = Entry(root, width=15)
result = Entry(root, width=15)

# image for empty
blank1 = Image.open("empty.jpg")
# resize image
resized_balnk1 = blank1.resize((150, 150), Image.ANTIALIAS)

empt1 = ImageTk.PhotoImage(resized_balnk1)
empty1 = Label(root, image=empt1)
empty1.image = empt1

# image for empty
blank2 = Image.open("empty2.jpg")
# resize image
resized_balnk2 = blank2.resize((150, 150), Image.ANTIALIAS)

empt2 = ImageTk.PhotoImage(resized_balnk2)
empty2 = Label(root, image=empt2)
empty2.image = empt2

loadNet.grid(row=0, column=0, padx=5, pady=5)
loadCitra.grid(row=1, column=0, padx=5, pady=5)
name.grid(row=2, column=0, padx=5, pady=5)
grayscale.grid(row=3, column=0, padx=5, pady=5)
identifikasi.grid(row=4, column=0, padx=5, pady=5)
result.grid(row=5, column=0, padx=5, pady=5)
reset.grid(row=6, column=0, padx=5, pady=5)

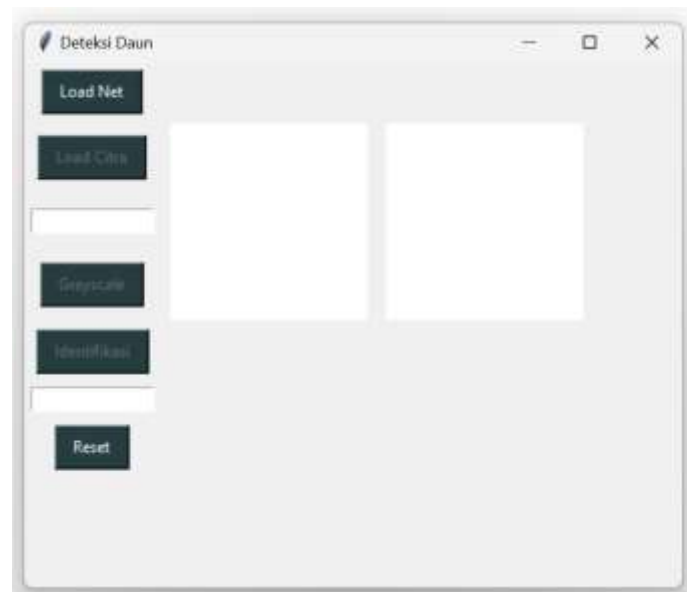
empty1.grid(row=1, column=1, rowspan=3, padx=5)
empty2.grid(row=1, column=2, rowspan=3, padx=5)

```

```
root.mainloop()
```

Pada potongan kode diatas menjelaskan pembuatan tampilan identifikasi daun diawali dengan pengimpor an library yang diperlukan, selanjutnya mengatur ukuran window yang akan terbuka, mengatur ukuran gambar yang diolah. Mendefinisikan kelas/kategori daun sesuai dengan urutan array (daun ganja, daun pepaya dan daun singkong) dan mengatur tampilan seperti beberapa button dan box. Selain itu, pada potongan kode diatas diberikan fungsi untuk mengubah gambar menjadi grayscale dan juga fungsi identifikasi citra daun.

#### D. Hasil Uji Program

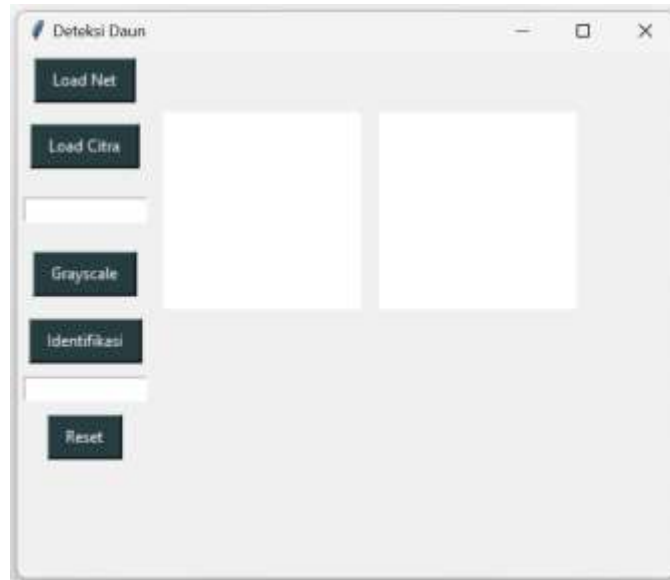


Gambar 15. Tampilan awal

Pada gambar 1. menunjukkan tampilan awal program deteksi daun ketika pertama kali dijalankan. Terlihat pada tampilan program ini terdapat beberapa button diantaranya:

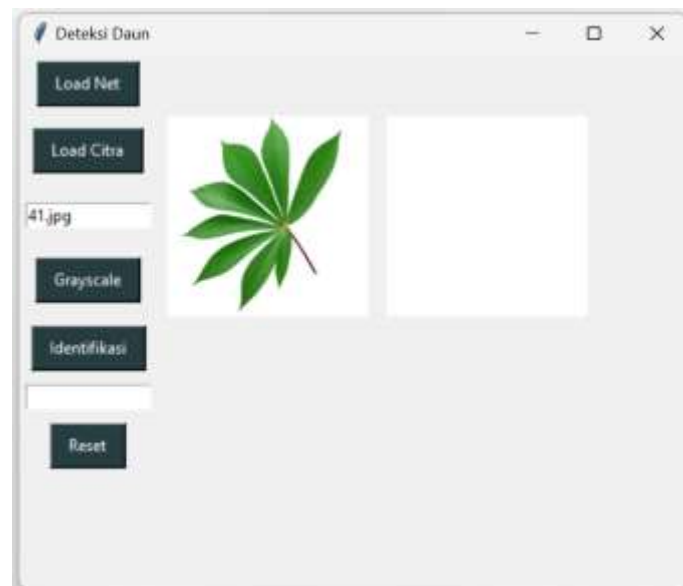
1. Load net
2. Load citra
3. Grayscale
4. Identifikasi
5. Reset

Selain itu, terdapat dua box/kotak putih yang nantinya digunakan untuk *meload* citra daun yang akan diidentifikasi.



Gambar 16. Tampilan load net

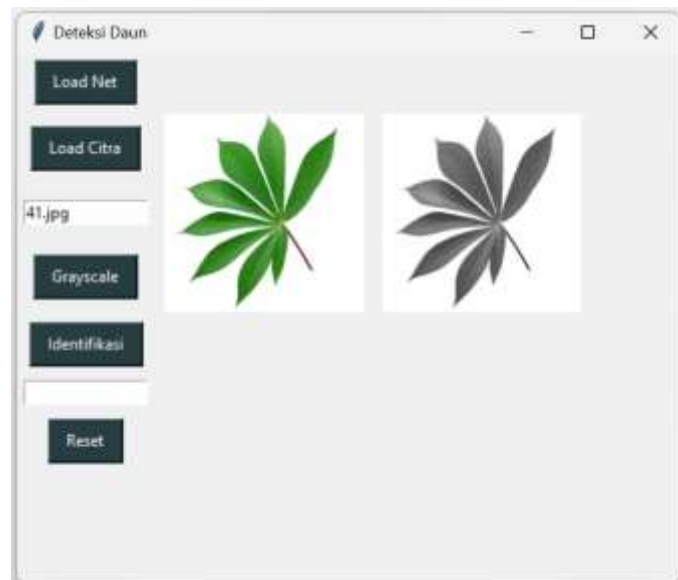
Pada gambar 2. menunjukkan tampilan ketika pengguna telah menekan button load net, pada kondisi ini semua button dapat berfungsi dengan baik (*non disabled*).



Gambar 17. Tampilan load citra

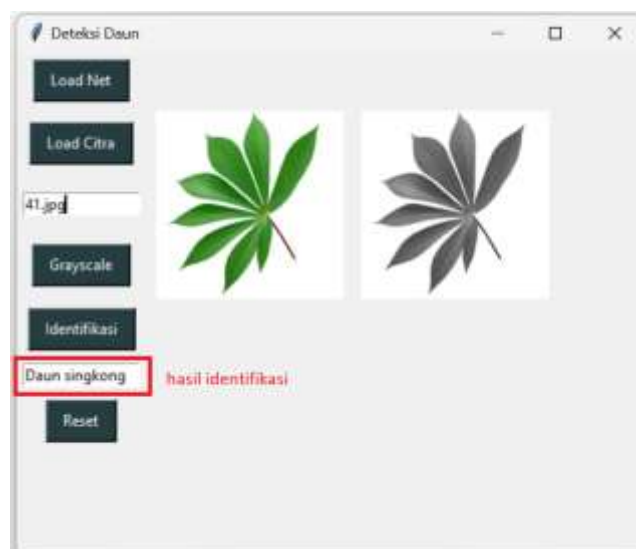
Pada gambar 3. menunjukkan tampilan program setelah pengguna menekan button load citra. Pada kondisi ini akan muncul pop up library pengguna dan pengguna dapat memilih gambar mana yang akan diidentifikasi, setelah itu gambar yang terpilih akan ditampilkan pada salah satu box/kotak sesuai pada gambar 3.





Gambar 18. Tampilan ubah citra ke grayscale

Pada gambar 4. menunjukkan tampilan setelah pengguna menekan button grayscale, hal ini bertujuan untuk mengubah citra yang awalnya RGB menjadi Grayscale. Setelah gambar berhasil diubah, gambar akan ditampilkan pada box/kotak yang tersedia.



Gambar 19. Tampilan hasil identifikasi citra

Pada gambar 5. menunjukkan tampilan setelah pengguna menekan button identifikasi. Pada kondisi ini, program akan otomatis melakukan pencocokan gambar yang ada di dataset kemudian memunculkan hasil identifikasi pada textbox yang tersedia sesuai pada gambar 5 diatas.

### **BAB III**

### **KESIMPULAN**

Program identifikasi citra daun menggunakan algoritma *convolutional neural network* (CNN) ini dapat digunakan untuk mengidentifikasi tiga kelas daun, di antaranya daun singkong, daun ganja dan daun pepaya. Dengan menggunakan algoritma *convolutional neural network* (CNN) didapatkan hasil akurasi sebesar 0.8095. Dengan dibuatnya program ini diharapkan dapat menjadi acuan dan sumber referensi pembelajaran untuk mahasiswa lainya.

## DAFTAR PUSTAKA

Kuriniawan, S. D. (2019). IDENTIFIKASI CITRA DAUN DENGAN MENGGUNAKAN METODE DEEP LEARNING CONVOLUTION NEURAL NETWORK (CNN).  
*Doctoral dissertation, Univeristas Balikpapan.*