**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](#_Toc122466233)

[DAFTAR ISI iii](#_Toc122466234)

[BAB I 1](#_Toc122466235)

[A. Deskripsi Program 1](#_Toc122466236)

[B. Dataset 1](#_Toc122466237)

[C. Tujuan 1](#_Toc122466238)

[BAB II 2](#_Toc122466239)

[A. Metodologi Penelitian 2](#_Toc122466240)

[B. Explore Data Analysis 3](#_Toc122466241)

[C. Deep Learning (Implementasi CNN) 9](#_Toc122466242)

[D. Hasil Uji Program 20](#_Toc122466243)

[BAB III 23](#_Toc122466244)

[DAFTAR PUSTAKA 24](#_Toc122466245)

# BAB I

**PENDAHULUAN**

## 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).

## 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.

## Tujuan

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

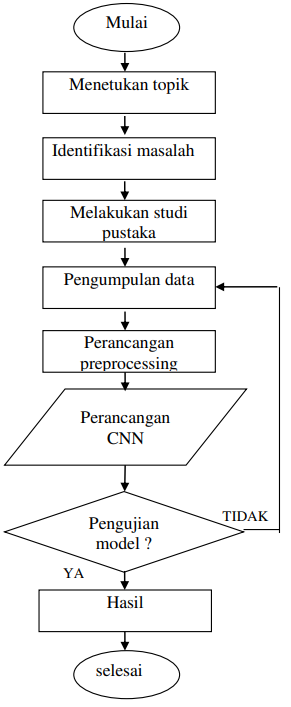
1. Mengetahui cara kerja dan hasil akurasi algoritma CNN terhadap citra daun
2. Melakukan identifikasi atau klasifikasi citra daun berdasarkan tiga kelas klasifikasi, yaitu daun singkong, daun ganja dan daun pepaya.

# 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:

## 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.

## 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:



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') |

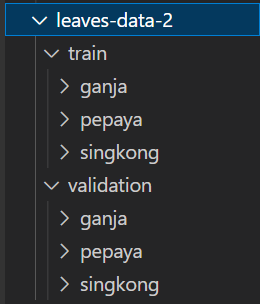
Pada potongan kode diatas menjelaskan pembuatan dua folder dengan nama “train” dan “validation” yang ada di dalam folder “leaves-data-2” sebelumnya.

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) |

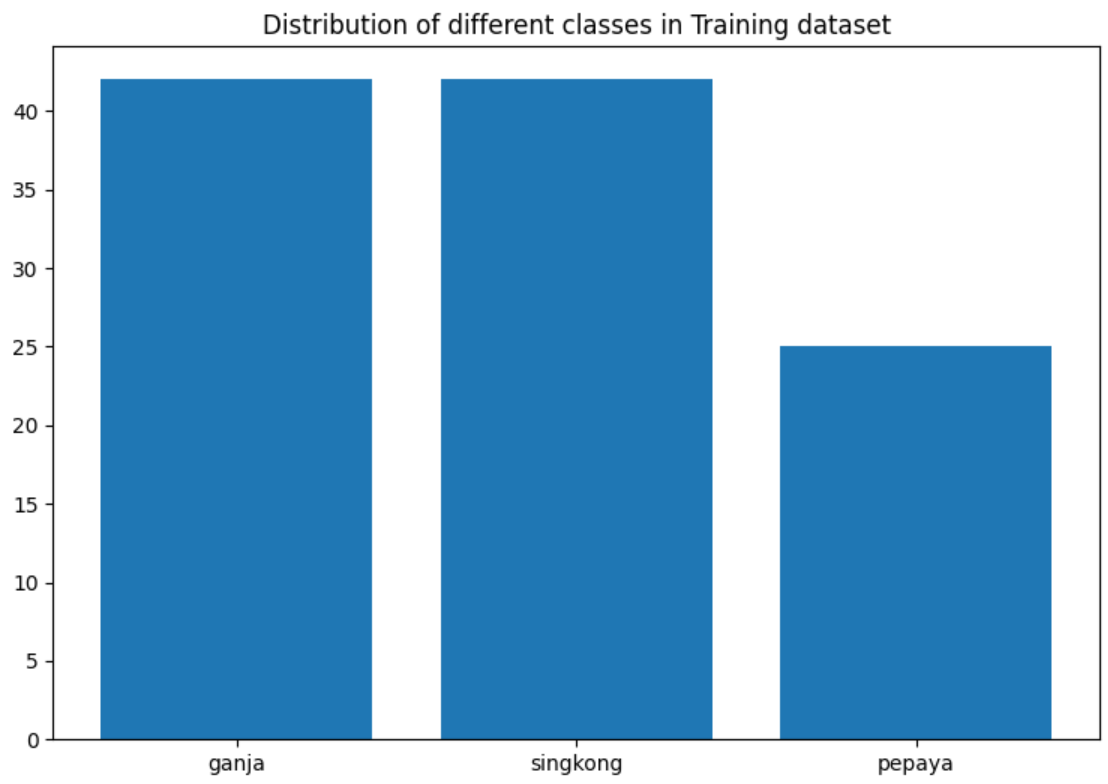
Pada potongan kode diatas menjelaskan melakukan split data sebanyak 0.85 dari keseluruhan data (total data, training data, dan validation data).

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 peng*import* an *library* terlebih dahulu kemudian mendefinisikan nama folder dan menampilkannya pada output.

Output:



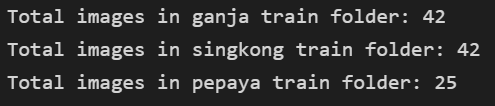
Gambar 4. Output hasil

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:



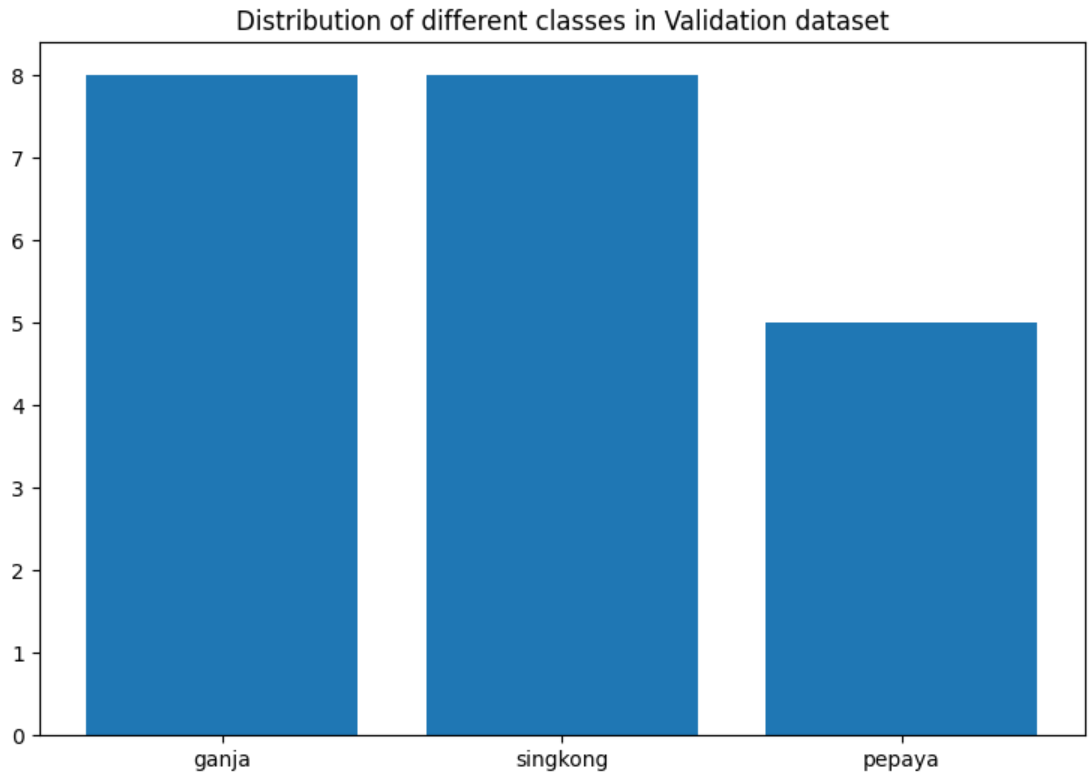
Gambar 5. Output hasil

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:



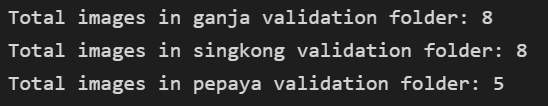
Gambar 6. Output hasil

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:



Gambar 7. Output hasil

## 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:



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:



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) |

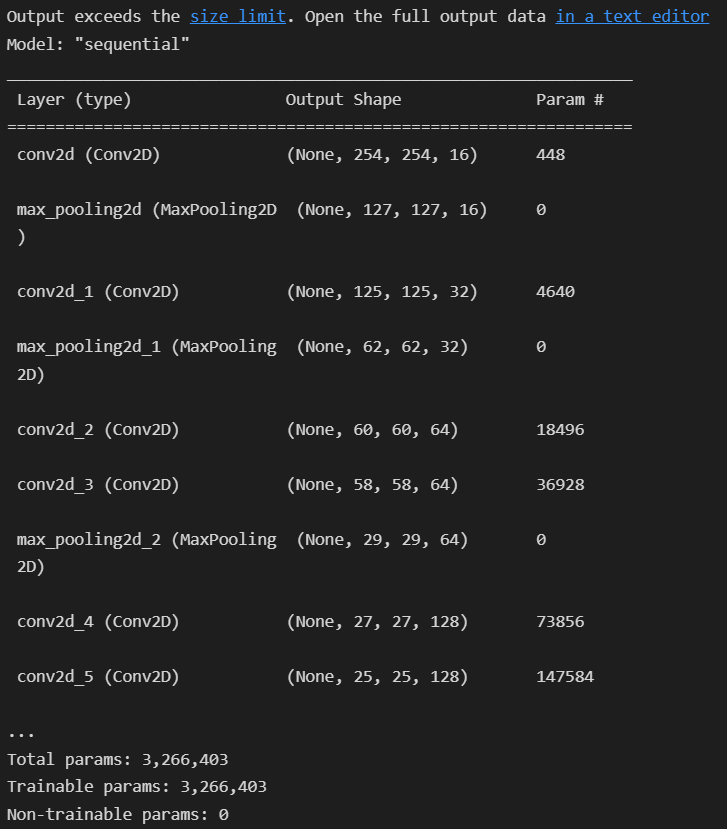
Pada potongan kode diatas menjelaskan cara menyimpan semua data yang telah di train dan validation sebagai file h5 dengan nama file “best\_model.h5”.

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:



Gambar 10. Output hasil

Kode:

|  |
| --- |
| model.compile(optimizer='Adam',  loss='categorical\_crossentropy',  metrics=['accuracy']) |

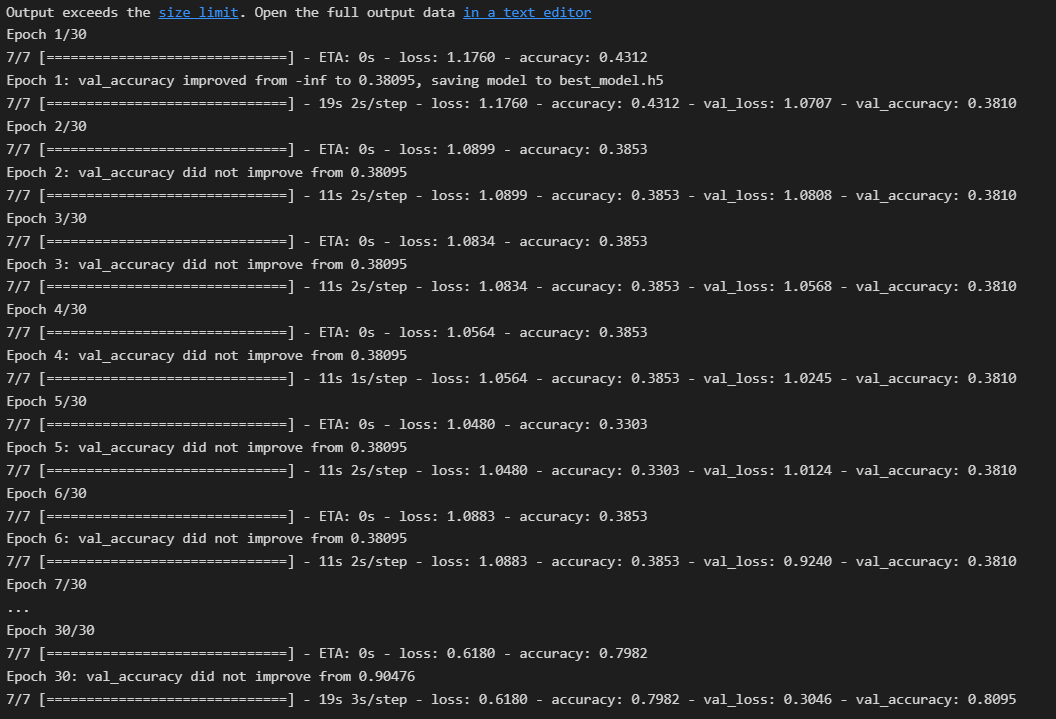
Pada potongan kode diatas menjelaskan pengoptimalan menggunakan adam.

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:



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:



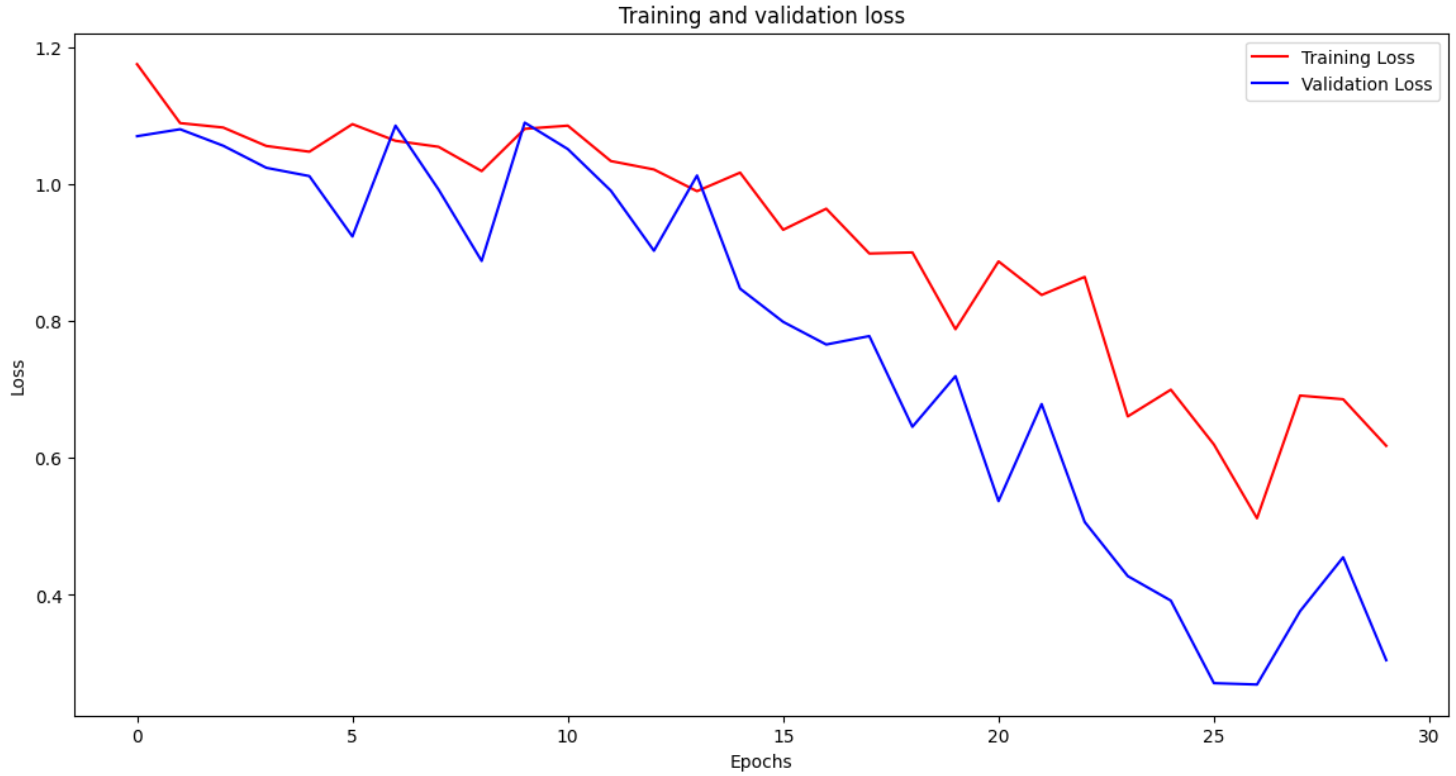
Gambar 12. Output hasil

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:



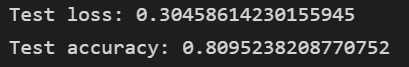
Gambar 13. Output hasil

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:



Gambar 14. Output hasil

Kode:

|  |
| --- |
| from keras import utils  from keras.models import load\_model  import numpy as np |

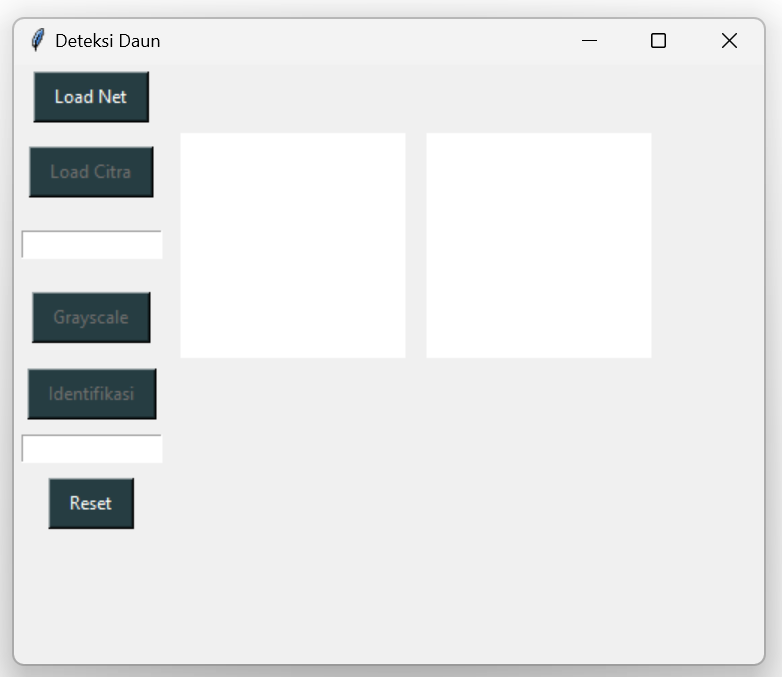
Pada potongan kode diatas menjelaskan peng*import* an *library* keras dan numpy.

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.filname = filedialog.askopenfilename(initialdir='D:/UPN/Semester/semester5', title="Select A File", filetypes=(("jpg files", "\*.jpg"), ("all files", "\*.\*")))  # load image  ori\_image = Image.open(root.filname)  # 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.filname.split("/")[-1]  # print file name in input  name.insert(0, file\_name)  # function grayscale image  def grayscale\_image():  gray\_img = Image.open(root.filname).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.filname).convert('L')  # resize image  img = utils.load\_img(root.filname, 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.

## Hasil Uji Program

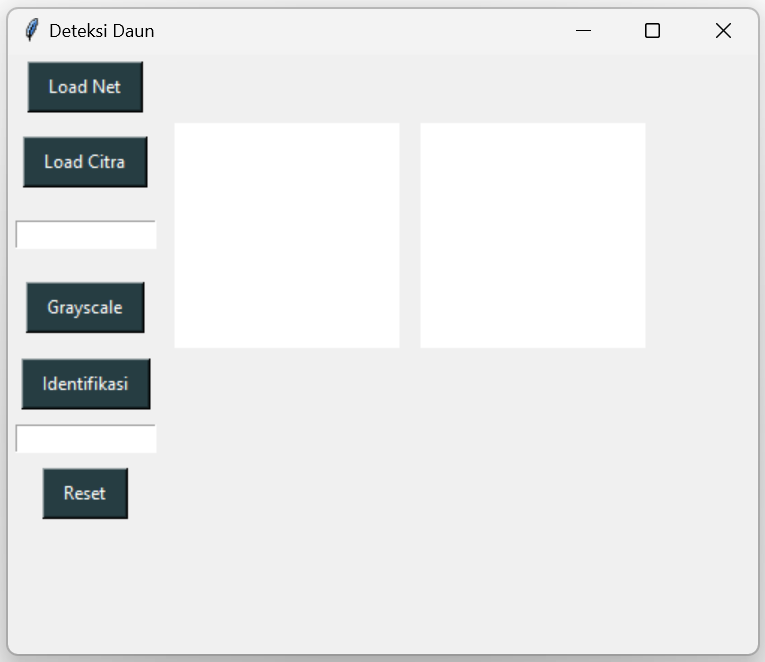


Gambar 15. Tampilan awal

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

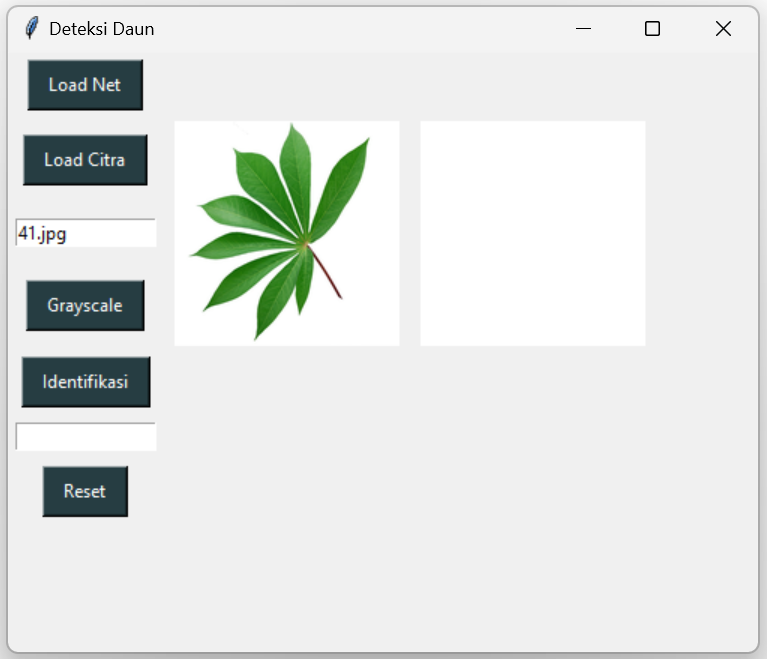
1. Load net
2. Load citra
3. Grayscale
4. Identifikasi
5. Reset

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



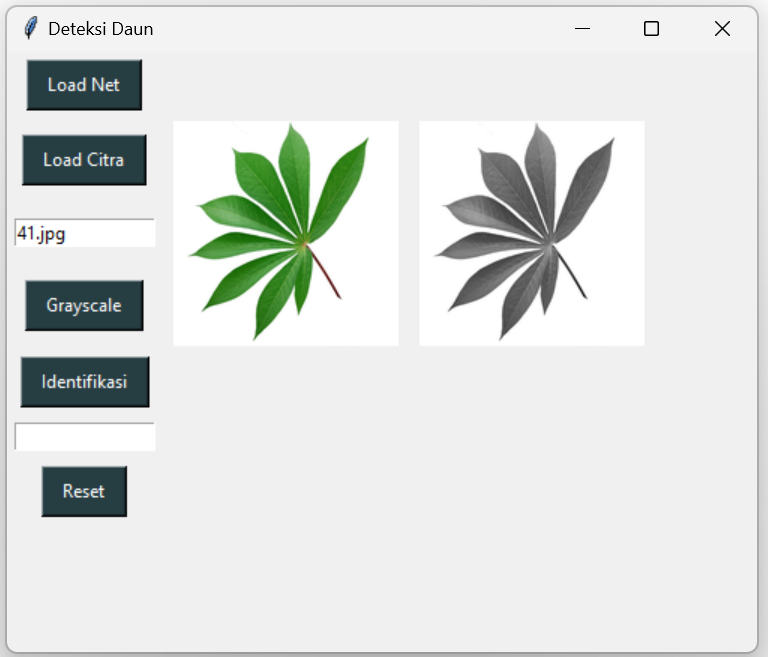
Gambar 16. Tampilan load net

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



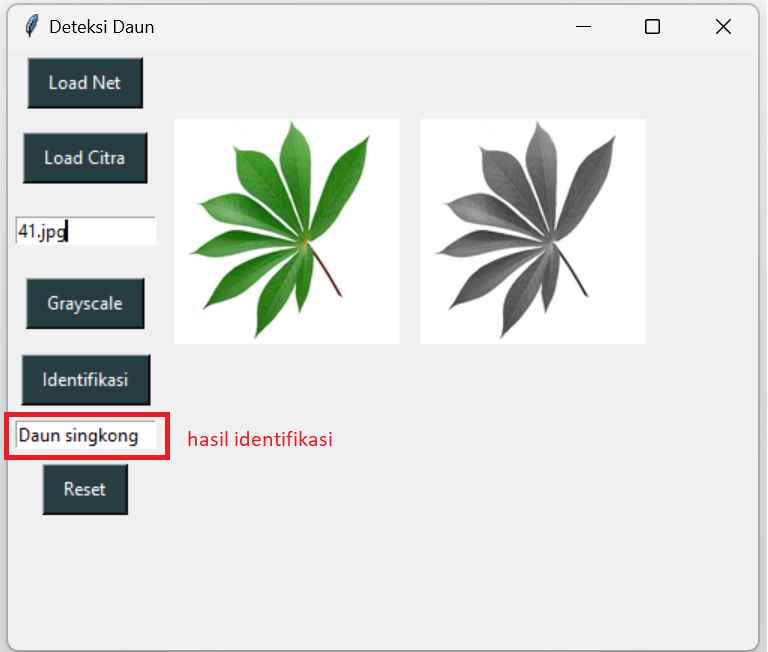
Gambar 17. Tampilan load citra

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 ditampilan 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*.