Kelompok : Pandas

Anggota :

- **Kevin Stevano Rianto / 220711655**
- **Averina Harseno Subianto / 220711757**
- **Christofer Nicholas Japri / 220711803**
- **Ivona Atha Nur Afiah / 220711805**

Arsitektur VGG-16:

```
import tensorflow as tf
import cv2
import numpy as np
from matplotlib import pyplot as plt
#load data
data_dir = r"D:\atma\Semester 5\ML\UAS\train_data"
#Randomize data yang telah di load sekaligus resize menjadi 180 x 180
data = tf.keras.utils.image_dataset_from_directory(data_dir,
seed=123, image_size=(224,224), batch_size=16)
print(data.class_names)

class_names = data.class_names
print("Class names: ", class_names)

img_size = 224
batch = 16
validation_split = 0.1
dataset = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
)
total_count = len(dataset)
val_count = int(total_count*validation_split)
train_count = total_count - val_count

print("Total images:", total_count)
print("Train images:", train_count)
print("Validation images:", val_count)

train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count)
```

```
import matplotlib.pyplot as plt

i = 0
plt.figure(figsize=(10,10))

#tampilkan untuk memastikan data sudah di load
for images, labels in train_ds.take(1):
    for i in range(9):
        plt.subplot(3,3, i+1)
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(class_names[labels[i]])
        plt.axis('off')
```

```
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)

#loop untuk mengecek atribut gambar(jumlah, tinggi, lebar, dan
channel(RGB))
```

```
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model

Tuner = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size =
Tuner)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = Tuner)

#Augmentasi data dengan menggunakan Sequential
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape = (img_size,
img_size, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])

i = 0
plt.figure(figsize=(10,10))
#Lihat data setelah di augmentasi
for images, labels in train_ds.take(69):
    for i in range(9):
        images = data_augmentation(images)
```

```
        plt.subplot(3,3, i+1)
        plt.imshow(images[0].numpy().astype('uint8'))
        plt.axis('off')
```

```
import tensorflow as tf
import keras

import keras._tf_keras.keras.backend as K
from keras._tf_keras.keras.models import Model
from keras._tf_keras.keras.layers import Input, Dense, Conv2D
from keras._tf_keras.keras.layers import Flatten, MaxPool2D
from keras._tf_keras.keras.layers import Dropout

def vgg16(input_shape, n_classes):
    input = Input(input_shape)

    x = Conv2D(64, (3, 3), padding='same', activation='relu')(input)
    x = Conv2D(64, (3, 3), padding='same', activation='relu')(x)
    x = MaxPool2D((2, 2), strides=(2, 2))(x)

    x = Conv2D(128, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(128, (3, 3), padding='same', activation='relu')(x)
    x = MaxPool2D((2, 2), strides=(2, 2))(x)

    x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)
    x = MaxPool2D((2, 2), strides=(2, 2))(x)

    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = MaxPool2D((2, 2), strides=(2, 2))(x)

    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = MaxPool2D((2, 2), strides=(2, 2))(x)

    x = Flatten()(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
```

```
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(n_classes, activation='softmax')(x)

    model = Model(input, output)
    return model

input_shape = (224, 224, 3)
n_classes = 3

K.clear_session()

model = vgg16(input_shape, n_classes)
model.summary()
```

```
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
#Coimpile dengan optimizer adam
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
#buat early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=7,
                               mode='max')
#fit validation data ke dalam model
history= model.fit(train_ds,
                   epochs=30,
                   validation_data=val_ds,
                   callbacks=[early_stopping])
```

```
#buat plot dengan menggunakan history supaya jumlahnya sesuai epoch
yang dilakukan
ephocs_range = range(1, len(history.history['loss']) + 1)
plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.plot(ephocs_range, history.history['accuracy'], label='Training
Accuracy')
```

```
plt.plot(ephocs_range, history.history['val_accuracy'],
label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(ephocs_range, history.history['loss'], label='Training
Loss')
plt.plot(ephocs_range, history.history['val_loss'], label='Validation
Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

```
model.save('BestModel_VGG-16_Pandas.h5')
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

# Load the trained model
model = load_model(r'D:\atma\Semester 5\ML\UAS\vgg16.h5')  # Ganti
dengan path model Anda
class_names = ['Aloevera', 'Echeveria', 'Sedum']

# Function to classify images and save the original image
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        # Load and preprocess the image
        input_image = tf.keras.utils.load_img(image_path,
target_size=(224, 224))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)  #
Add batch dimension

        # Predict
        predictions = model.predict(input_image_exp_dim)
        result = tf.nn.softmax(predictions[0])
        class_idx = np.argmax(result)
```

```
        confidence = np.max(result) * 100

        # Display prediction and confidence in notebook
        print(f"Prediksi: {class_names[class_idx]}")
        print(f"Confidence: {confidence:.2f}%")

        # Save the original image (without text)
        input_image = Image.open(image_path)
        input_image.save(save_path)

        return f"Prediksi: {class_names[class_idx]} dengan confidence
{confidence:.2f}%. Gambar asli disimpan di {save_path}."
    except Exception as e:
        return f"Terjadi kesalahan: {e}"

# Contoh penggunaan fungsi
result =
classify_images(r'test_data\Echeveria\2d8390bc-028e-430c-a774-52f02ab
67199.jpg', save_path='hasil2.jpg')
print(result)
```

```
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

# Muat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',  # Menghasilkan label dalam bentuk
one-hot encoding
    batch_size=32,
    image_size=(224, 224)
)

# Prediksi model
y_pred = model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)  # Konversi ke kelas
prediksi
```

```
# Ekstrak label sebenarnya dari test_data dan konversi ke bentuk
indeks kelas
true_labels = []
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())  # Konversi
one-hot ke indeks kelas
true_labels = tf.convert_to_tensor(true_labels)

# Membuat matriks kebingungan
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

# Menghitung akurasi
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)

# Menghitung presisi dan recall
precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=1)

# Menghitung F1 Score
f1_score = 2 * (precision * recall) / (precision + recall)

# Visualisasi Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=['Aloevera', 'Echeveria', 'Sedum'],
yticklabels=['Aloevera', 'Echeveria', 'Sedum'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Menampilkan hasil
print("Confusion Matrix:\n", conf_mat.numpy())
print("Akurasi:", accuracy.numpy())
print("Presisi:", precision.numpy())
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())
```

Arsitektur AlexNet:

```
#Import library
import os
import numpy as np

#Import library tensorflow dan modul keras yang diperlukan
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import load_img,
ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
Dropout, Flatten

#Penjelasan
# layers digunakan untuk menambahkan lapisan ke dalam model
# load_img digunakan untuk memuat gambar
# ImageDataGenerator digunakan untuk melakukan augmentasi pada gambar
# Sequential digunakan untuk membuat model secara berurutan
# Conv2D digunakan untuk membuat lapisan konvolusi
# MaxPooling2D digunakan untuk melakukan pooling pada lapisan
konvolusi
# Dense digunakan untuk membuat lapisan fully connected
# Dropout digunakan untuk menghindari overfitting
# Flatten digunakan untuk membuat lapisan menjadi flat (rata) menjadi
vektor 1 dimensi
```

```
count = 0 #digunakan untuk menghitung jumlah gambar
dirs = os.listdir(r"C:\Users\H
P\OneDrive\Documents\mldl\UAS\train_data")
for dir in dirs:
    files = list(os.listdir(r"C:\Users\H
P\OneDrive\Documents\mldl\UAS\train_data/"+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
# Parameter
base_dir = r"C:\Users\H P\OneDrive\Documents\mldl\UAS\train_data"
```

```
img_size = 180 #mengubah ukuran gambar menjadi 180
batch = 16 #jumlah sample (gambar) yang akan diproses pada satu kali iterasi
validation_split = 0.1 #data pelatihan yang akan digunakan sebagai data validasi
```

```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir, #path direktori, subfolder dianggap sebagai label
    seed=123, #untuk memastikan proses pemisahan data selalu konsisten (random_state)
    image_size=(img_size, img_size), #ukuran gambar diubah (resize) menjadi 180x180 pixel
    batch_size=batch, #jumlah gambar yang akan dikelompokkan
)
```

```
#mendapatkan nama kelas dari dataset
class_names = dataset.class_names #dataset.class_names akan mengambil daftar nama kelas berdasarkan subfolder di dalam direktori
print("Class Names:", class_names)
```

```
total_count = len(dataset) #menghitung jumlah total gambar dalam dataset
val_count = int(total_count * validation_split) #menghitung jumlah gambar untuk validasi
train_count = total_count - val_count # menghitung jumlah gambar untuk train

print("Total Images:", total_count)
print("Train Images:", train_count)
print("Validation Images:", val_count)
```

```
train_ds = dataset.take(train_count) #digunakan untuk mengambil (take) sejumlah batch sebanyak 'train count' yang pertama dari data set
val_ds = dataset.skip(train_count) #digunakan untuk melewati (skip) sejumlah batch sebanyak 'train count' yang pertama dari data set
```

```
import matplotlib.pyplot as plt

i = 0
plt.figure(figsize=(10,10)) #membuat figure dengan ukuran 10x10 inchi untuk menampilkan gambar

for images, labels in train_ds.take(1): #mengambil 1 batch pertama
```

```
dari train_ds
    for i in range(9): #iterasi untuk menampilkan 9 hambar pertama
dalam batch
        plt.subplot(3,3, i+1) #menyiapkan subplot dengan grid 3x3 dan
menempatkan gambar pada posisi i+1
        plt.imshow(images[i].numpy().astype('uint8')) #menampilkan
gambar dan mengonversi ke tipe uint8
        plt.title(class_names[labels[i]]) #menampilkan judul gambar
sesuai dengan nama kelas
        plt.axis('off') #menonaktifkan sumbu pada gambar agar tidak
terlihat
```

```
import numpy as np

# Tampilkan gambar dengan shape (16, 180, 180, 3)
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)  # Output: (16, 180, 180, 3)
    #16: Jumlah gambar dalam batch.
    #180: Lebar gambar dalam piksel
    #180: Tinggi gambar dalam piksel
    #3: Jumlah channel gambar (RGB)
```

```
#Mengatur AUTOTUNE untuk pemrosesan data otomatis oleh tensorflow
#AUTOTUNE digunakan untuk memungkinkan tensorflow mengoptimalkan
jumlah thread secara otomatis saat memproses data
AUTOTUNE = tf.data.AUTOTUNE
```

```
#mengoptimalkan dataset pelatihan (train_ds)
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size =
AUTOTUNE)
#cache digunakan untuk menyimpan dataser di memori agar lebih cepat
diakses
#shuffle mengacak data dalam batch agar model tidak terlalu terlatih
pada urutan tertentu
#prefetch untuk menyiapkan data batch berikutnya secara otomatis
```

```
#mengoptimalkan dataset validasi (val_ds)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size =
AUTOTUNE)
```

```
data_augmentation = Sequential([
```

```
    layers.RandomFlip("horizontal", input_shape =
(img_size,img_size,3)), #membalik gambar secara horizontal
    layers.RandomRotation(0.1), #merotasi gambar secara acak dalam
kisaran 0°-36° (0.1 * 360)
    layers.RandomZoom(0.1) #melakukan zoom in/zoom out secara acak
dengan rentang 10%
])
```

```
#sama seperti sebelumnya, code ini digunakan untuk menampilkan gambar
dari data_augmentation
i = 0
plt.figure(figsize=(10,10))

for images, labels in train_ds.take(1):
    for i in range(9):
        images = data_augmentation(images)
        plt.subplot(3,3, i+1)
        plt.imshow(images[0].numpy().astype('uint8'))
        plt.axis('off')
```

```
import tensorflow as tf
import keras

import keras._tf_keras.keras.backend as K
from keras._tf_keras.keras.models import Model
from keras._tf_keras.keras.layers import Input, Dense, Conv2D
from keras._tf_keras.keras.layers import Flatten, MaxPool2D,
AvgPool2D
from keras._tf_keras.keras.layers import Concatenate, Dropout,
BatchNormalization

from keras._tf_keras.keras.models import load_model

#membuat model from scratch
def alexnet(input_shape, n_classes):
    input = Input(input_shape)

    x = Conv2D(96, kernel_size=(11, 11), strides=(4, 4),
padding='valid', activation='relu')(input)
    x = MaxPool2D(pool_size=(3, 3), strides=(2, 2),
padding='valid')(x)
```

```
    x = BatchNormalization()(x)

    x = Conv2D(256, kernel_size=(5, 5), strides=(1, 1),
padding='same', activation='relu')(x)
    x = MaxPool2D(pool_size=(3, 3), strides=(2, 2),
padding='valid')(x)
    x = BatchNormalization()(x)

    x = Conv2D(384, kernel_size=(3, 3), strides=(1, 1),
padding='same', activation='relu')(x)
    x = BatchNormalization()(x)

    x = Conv2D(384, kernel_size=(3, 3), strides=(1, 1),
padding='same', activation='relu')(x)
    x = BatchNormalization()(x)

    x = Conv2D(256, kernel_size=(3, 3), strides=(1, 1),
padding='same', activation='relu')(x)
    x = MaxPool2D(pool_size=(3, 3), strides=(2, 2),
padding='valid')(x)
    x = BatchNormalization()(x)

    x = Flatten()(x)

    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.4)(x)
    x = BatchNormalization()(x)

    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.4)(x)
    x = BatchNormalization()(x)

    output = Dense(n_classes, activation='softmax')(x)

    model = Model(input, output)
    return model

#Pastikan input shae dan jumlah kelas sesuai
input_shape = (180, 180, 3)
n_classes = 3

#Clear Cache Keras menggunakan clear session
```

```
K.clear_session()
#buat model dengan
model = alexnet(input_shape, n_classes)
model.summary()
```

```
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
#Coimpile dengan optimizer adam
model.compile(
    optimizer=Adam(learning_rate=0.00001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

#buat early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=5,
                               mode='max')
#fit validation data ke dalam model
history= model.fit(train_ds,
                   epochs=30,
                   validation_data=val_ds,
                   callbacks=[early_stopping])
```

```
#buat plot dengan menggunakan history supaya jumlahnya sesuai epoch
yang dilakukan
ephocs_range = range(1, len(history.history['loss']) + 1)
plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.plot(ephocs_range, history.history['accuracy'], label='Training
Accuracy')
plt.plot(ephocs_range, history.history['val_accuracy'],
label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(ephocs_range, history.history['loss'], label='Training
Loss')
plt.plot(ephocs_range, history.history['val_loss'], label='Validation
Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
```

```
plt.show()
```

```
model.save('alexnet.h5')
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

#memuat model yang sudah dilatih
model = load_model(r"C:\Users\H
P\OneDrive\Documents\mldl\UAS\alexnet.h5")  # Ganti dengan path model
Anda
class_names = ["Aloevera", "Echeveria", "Sedum"] #kelas yang ada pada
model

#fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        #memuat dan mempersiapkan gambar untuk prediksi
        input_image = tf.keras.utils.load_img(image_path,
target_size=(180, 180)) #membuat gambar dari path dan mnegubah
ukurannya menjadi 180x180 pixel
        input_image_array = tf.keras.utils.img_to_array(input_image)
#mengubah gambar jadi array numpy agar bisa di proses model
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)
#menambahkan dimensi batch agar sesuai dengan input model

#dimensi menjadi (1, 180, 180, 3)

        #melakukan prediksi
        predictions = model.predict(input_image_exp_dim) #melakukan
prediksi pada gambar yang telah diproses
        result = tf.nn.softmax(predictions[0]) #menghitung hasil
prediksi menggunakan softmax untuk mendapatkan probabilitas tiap
kelas
        class_idx = np.argmax(result) #menemukan indeks kelas dengan
probabilitas tertinggi
        confidence = np.max(result) * 100 #menghitung confidence
dalam persentase

        #menampilkan hasil prediksi dan confidence
```

```
        print(f"Prediksi: {class_names[class_idx]}") #menampilkan
nama kelas yang diprediksi
        print(f"Confidence: {confidence:.2f}%") #menampilkan nilai
confidence

        #menyimpan gambar asli tanpa teks
        input_image = Image.open(image_path) #membuka gambar yang ada
di path
        input_image.save(save_path) #menyimpan gambar asli ke dalam
path yang telah ditentukan

        return f"Prediksi: {class_names[class_idx]} dengan confidence
{confidence:.2f}%. Gambar asli disimpan di {save_path}."
    except Exception as e:
        return f"Terjadi kesalahan: {e}"

#contoh penggunaan fungsi
result = classify_images(r"C:\Users\H
P\OneDrive\Documents\mldl\UAS\test_data\Echeveria\Echeveria.jpg",
save_path='Echeveria.jpg')
print(result)
```

```
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

# Muat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r"C:\Users\H P\OneDrive\Documents\mldl\UAS\test_data",
    labels='inferred',
    label_mode='categorical',  # Menghasilkan label dalam bentuk
one-hot encoding
    batch_size=32,
    image_size=(180, 180)
)

# Prediksi model
y_pred = model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)  # Konversi ke kelas
prediksi
```

```
# Ekstrak label sebenarnya dari test_data dan konversi ke bentuk
indeks kelas
true_labels = []
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())  # Konversi
one-hot ke indeks kelas
true_labels = tf.convert_to_tensor(true_labels)

# Membuat matriks kebingungan
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

# Menghitung akurasi
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)

# Menghitung presisi dan recall
precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=1)

# Menghitung F1 Score
f1_score = 2 * (precision * recall) / (precision + recall)

# Visualisasi Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Aloevera", "Echeveria", "Sedum"],
yticklabels=["Aloevera", "Echeveria", "Sedum"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Menampilkan hasil
print("Confusion Matrix:\n", conf_mat.numpy())
print("Akurasi:", accuracy.numpy())
print("Presisi:", precision.numpy())
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())
```

GOOGLENET

```python
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\ASUS\Documents\2024\ML\UAS_Googlenet\testing',
    labels='inferred',
    label_mode='categorical',
    batch_size=10,
    image_size=(224, 224)
)

y_pred = model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)

true_labels = []
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())
true_labels = tf.convert_to_tensor(true_labels)

conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) / tf.reduce_sum(conf_mat)

precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat, axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat, axis=1)

precision = tf.where(tf.math.is_nan(precision), tf.zeros_like(precision), precision)
recall = tf.where(tf.math.is_nan(recall), tf.zeros_like(recall), recall)

f1_score = 2 * (precision * recall) / (precision + recall)
f1_score = tf.where(tf.math.is_nan(f1_score), tf.zeros_like(f1_score), f1_score)

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Aloevera", "Echeveria", "Sedum"], yticklabels=["Aloevera", "
Echeveria", "Sedum"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

print("Confusion Matrix:\n", conf_mat.numpy())
print("Accuracy:", accuracy.numpy())
print("Precision:", precision.numpy())
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())

```

```python
import matplotlib.pyplot as plt

i = 0
plt.figure(figsize=(10,10))


for images, labels in train_ds.take(1):
    for i in range(9):
        plt.subplot(3,3, i+1)
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(class_names[labels[i]])
        plt.axis('off')
```

```python
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

```python
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model

Tuner = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = Tuner)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = Tuner)


data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape = (img_size,img_size,3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])

i = 0
plt.figure(figsize=(10,10))
for images, labels in train_ds.take(69):
    for i in range(9):
        images = data_augmentation(images)
        plt.subplot(3,3, i+1)
        plt.imshow(images[0].numpy().astype('uint8'))
        plt.axis('off')
```

```python
import tensorflow as tf
import keras

import keras._tf_keras.keras.backend as K
from keras._tf_keras.keras.models import Model
from keras._tf_keras.keras.layers import Input, Dense, Conv2D
from keras._tf_keras.keras.layers import Flatten, MaxPool2D, AvgPool2D
from keras._tf_keras.keras.layers import Concatenate, Dropout

from keras._tf_keras.keras.models import load_model

def googlenet(input_shape, n_classes):

    def inception_block(x, f):
        t1 = Conv2D(f[0], 1, activation='relu')(x)

        t2 = Conv2D(f[1], 1, activation='relu')(x)
        t2 = Conv2D(f[2], 3, padding='same', activation='relu')(t2)

        t3 = Conv2D(f[3], 1, activation='relu')(x)
        t3 = Conv2D(f[4], 5, padding='same', activation='relu')(t3)

        t4 = MaxPool2D(3, 1, padding='same')(x)
        t4 = Conv2D(f[5], 1, activation='relu')(t4)

        output = Concatenate()([t1, t2, t3, t4])
        return output


    input = Input(input_shape)

    x = Conv2D(64, 7, strides=2, padding='same', activation='relu')(input)
    x = MaxPool2D(3, strides=2, padding='same')(x)

    x = Conv2D(64, 1, activation='relu')(x)
    x = Conv2D(192, 3, padding='same', activation='relu')(x)
    x = MaxPool2D(3, strides=2)(x)

    x = inception_block(x, [64, 96, 128, 16, 32, 32])
    x = inception_block(x, [128, 128, 192, 32, 96, 64])
    x = MaxPool2D(3, strides=2, padding='same')(x)

    x = inception_block(x, [192, 96, 208, 16, 48, 64])
    x = inception_block(x, [160, 112, 224, 24, 64, 64])
    x = inception_block(x, [128, 128, 256, 24, 64, 64])
    x = inception_block(x, [112, 144, 288, 32, 64, 64])
    x = inception_block(x, [256, 160, 320, 32, 128, 128])
    x = MaxPool2D(3, strides=2, padding='same')(x)

    x = inception_block(x, [256, 160, 320, 32, 128, 128])
    x = inception_block(x, [384, 192, 384, 48, 128, 128])

    x = AvgPool2D(3, strides=1)(x)
    x = Dropout(0.4)(x)

    x = Flatten()(x)
    output = Dense(n_classes, activation='softmax')(x)

    model = Model(input, output)
    return model

input_shape = 224, 224, 3
n_classes = 3

K.clear_session()
model = googlenet(input_shape, n_classes)
model.summary()

```

```python
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

model.compile(
    optimizer=Adam(learning_rate=5e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=5,
                               mode='max')

history= model.fit(train_ds,
                   epochs=54,
                   validation_data=val_ds,
                   callbacks=[early_stopping])
```

```python
ephocs_range = range(1, len(history.history['loss']) + 1)
plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.plot(ephocs_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(ephocs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(ephocs_range, history.history['loss'], label='Training Loss')
plt.plot(ephocs_range, history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

```
model.save('googlenet.h5')
```

```python
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image

model = load_model(r'C:\Users\ASUS\Documents\2024\ML\UAS_Googlenet\googlenet.h5')
class_names = ['Aloevera', 'Echeveria', 'Sedum']

def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        # Load and preprocess the image
        input_image = tf.keras.utils.load_img(image_path, target_size=(224, 224))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_array /= 255.0
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)

        predictions = model.predict(input_image_exp_dim)
        result = tf.nn.softmax(predictions[0])
        class_idx = np.argmax(result)
        confidence = np.max(result) * 100

        print(f"Prediksi: {class_names[class_idx]}")
        print(f"Confidence: {confidence:.2f}%")

        input_image = Image.open(image_path)
        input_image.save(save_path)

        return f"Prediksi: {class_names[class_idx]} dengan confidence {confidence:.2f}%. Gambar asli disimpan di {save_path}."
    except Exception as e:
        return f"Terjadi kesalahan: {e}"

result = classify_images(r'C:\Users\ASUS\Documents\2024\ML\UAS_Googlenet\testing\Sedum\IMG_8077.JPG', save_path='sedum.jpg')
print(result)
```

```python
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\ASUS\Documents\2024\ML\UAS_Googlenet\testing',
    labels='inferred',
    label_mode='categorical',
    batch_size=10,
    image_size=(224, 224)
)

y_pred = model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)

true_labels = []
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())
true_labels = tf.convert_to_tensor(true_labels)

conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) / tf.reduce_sum(conf_mat)

precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat, axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat, axis=1)

precision = tf.where(tf.math.is_nan(precision), tf.zeros_like(precision), precision)
recall = tf.where(tf.math.is_nan(recall), tf.zeros_like(recall), recall)

f1_score = 2 * (precision * recall) / (precision + recall)
f1_score = tf.where(tf.math.is_nan(f1_score), tf.zeros_like(f1_score), f1_score)

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Aloevera", "Echeveria", "Sedum"], yticklabels=["Aloevera", "
Echeveria", "Sedum"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

print("Confusion Matrix:\n", conf_mat.numpy())
print("Accuracy:", accuracy.numpy())
print("Precision:", precision.numpy())
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())

```

MobileNet

```python
#Import library
import os
import numpy as np
```

```
#Import library tensorflow dan modul keras yang diperlukan
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import load_img,
ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
Dropout, Flatten

#Penjelasan
# layers digunakan untuk menambahkan lapisan ke dalam model
# load_img digunakan untuk memuat gambar
# ImageDataGenerator digunakan untuk melakukan augmentasi pada gambar
# Sequential digunakan untuk membuat model secara berurutan
# Conv2D digunakan untuk membuat lapisan konvolusi
# MaxPooling2D digunakan untuk melakukan pooling pada lapisan
konvolusi
# Dense digunakan untuk membuat lapisan fully connected
# Dropout digunakan untuk menghindari overfitting
# Flatten digunakan untuk membuat lapisan menjadi flat (rata) menjadi
vektor 1 dimensi
```

```
count = 0 #digunakan untuk menghitung jumlah gambar
dirs = os.listdir(r'C:\Users\kevst\OneDrive\Documents\Kuliah\Sem
5\Pembelajaran Mesin dan Pembelajaran Mendalam\UAS\Dataset')
for dir in dirs:
    files =
list(os.listdir(r'C:\Users\kevst\OneDrive\Documents\Kuliah\Sem
5\Pembelajaran Mesin dan Pembelajaran Mendalam\UAS\Dataset/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
# Parameter
base_dir = r'C:\Users\kevst\OneDrive\Documents\Kuliah\Sem
5\Pembelajaran Mesin dan Pembelajaran Mendalam\UAS\Dataset'
#direktori folder dataset
img_size = 180 #mengubah ukuran gambar menjadi 180
batch = 10 #jumlah sample (gambar) yang akan diproses pada satu kali
```

```
iterasi
validation_split = 0.1 #data pelatihan yang akan digunakan sebagai
data validasi
```

```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir, #path direktori, subfolder dianggap sebagai label
    seed=123, #untuk memastikan proses pemisahan data selalu
konsisten (random_state)
    image_size=(img_size, img_size), #ukuran gambar diubah (resize)
menjadi 180x180 pixel
    batch_size=batch, #jumlah gambar yang akan dikelompokkan
)
```

```
#mendapatkan nama kelas dari dataset
class_names = dataset.class_names #dataset.class_names akan mengambil
daftar nama kelas berdasarkan subfolder di dalam direktori
print("Class Names:", class_names)
```

```
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
menghitung jumlah gambar untuk train
total_count = len(dataset)
val_count = int(total_count * validation_split)
train_count = total_count - val_count

print("Total Images:", total_count)
print("Train Images:", train_count)
print("Validation Images:", val_count)
```

```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count)
```

```
import matplotlib.pyplot as plt

i = 0
plt.figure(figsize=(10,10)) #membuat figure dengan ukuran 10x10 inchi
untuk menampilkan gambar

###Terdapat code yang hilang disini! lihat modul untuk menemukanya
for images, labels in train_ds.take(1): #mengambil 1 batch pertama
dari train_ds
```

```
    for i in range(9):
        plt.subplot(3,3, i+1) #menyiapkan subplot dengan grid 3x3 dan menempatkan gambar pada posisi i+1
        plt.imshow(images[i].numpy().astype('uint8')) #menampilkan gambar dan mengonversi ke tipe uint8
        plt.title(class_names[labels[i]]) #menampilkan judul gambar sesuai dengan nama kelas
        plt.axis('off') #menonaktifkan sumbu pada gambar agar tidak terlihat
```

```
import numpy as np

# Tampilkan gambar dengan shape (32, 180, 180, 3)
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)  # Output: (32, 180, 180, 3)
    #32: Jumlah gambar dalam batch.
    #180: Lebar gambar dalam piksel
    #180: Tinggi gambar dalam piksel
    #3: Jumlah channel gambar (RGB)
```

```
#Mengatur AUTOTUNE untuk pemrosesan data otomatis oleh tensorflow
#AUTOTUNE digunakan untuk memungkinkan tensorflow mengoptimalkan jumlah thread secara otomatis saat memproses data
AUTOTUNE = tf.data.AUTOTUNE
```

```
#mengoptimalkan dataset pelatihan (train_ds)
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
#cache digunakan untuk menyimpan dataser di memori agar lebih cepat diakses
#shuffle mengacak data dalam batch agar model tidak terlalu terlatih pada urutan tertentu
#prefetch untuk menyiapkan data batch berikutnya secara otomatis
```

```
#mengoptimalkan dataset validasi (val_ds)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

```
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape =
(img_size,img_size,3)), #membalik gambar secara horizontal
    layers.RandomRotation(0.1), #merotasi gambar secara acak dalam
kisaran 0°-36° (0.1 * 360)
    layers.RandomZoom(0.1) #melakukan zoom in/zoom out secara acak
dengan rentang 10%
])
```

```
#sama seperti sebelumnya, code ini digunakan untuk menampilkan gambar
dari data_augmentation
i = 0
plt.figure(figsize=(10,10))

for images, labels in train_ds.take(1):
    for i in range(9):
        images = data_augmentation(images)
        plt.subplot(3,3, i+1)
        plt.imshow(images[0].numpy().astype('uint8'))
        plt.axis('off')
```

```
#import library yang dibutuhkan
from tensorflow.keras.applications import MobileNet #digunakan untuk
memanfaatkan model yang sudah dilatih sebelumnya untuk pengenalan
gambar
from tensorflow.keras.models import Model #digunakan untuk membuat
dan mengonfigurasi arsitektur model

#membuat model dengan bobot yang telah dilatih sebelumnya
#include_top=False berarti tidak menggunakan lapisan klasifikasi dari
mobilenet hanya bagian ekstraksi fitur
base_model = MobileNet(include_top=False, input_shape = (img_size,
img_size, 3))

#membuka (unfreeze beberapa lapisan untuk proses fine tuning)
base_model.trainable = True #seluruh model bisa dilatih
fine_tune_at = len(base_model.layers) // 2  #menentukan bahwa
setengah lapisan terakhir akan di unfreeze
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False #mengunci (freeze) lapisan pertama hingga
setengah bagian pertama agar tidak dilatih kembali
```

```
#membuat model akhir dengan lapisan tambahan
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    base_model,
    layers.GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(len(class_names), activation='softmax')
])
```

```
from tensorflow.keras.optimizers import Adam #untuk mengoptimalkan proses pelatihan model

#mengkompilasi model dengan optimizer, loss function, dan metrics
model.compile(
    optimizer=Adam(learning_rate=1e-4), #menggunakan optimizer Adam dengan learning rate 0.0001
    loss='sparse_categorical_crossentropy', #untuk klasifikasi multi-kelas
    metrics=['accuracy'] #akurasi digunakan sebagai metrik evaluasi
)
```

```
#menampilkan ringkasan dari model
model.summary()
```

```
#early stopping digunakan untuk menghentikan pelatihan lebih awal jika model tidak ada peningkatan
from tensorflow.keras.callbacks import EarlyStopping

#Ada fungsi early stopping disini, jangan keskip tuan :D
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=3,
                               mode='max')

#melatih model menggunakan data latih dan validasi dengan early
```

```
stopping
history= model.fit(train_ds, #data pelatihan yang telah disiapkan
                   epochs=30, # jumlah maksimal epoch
                   validation_data=val_ds,  #data validasi untuk
mengevaluasi model pada setiap epoch
                   callbacks=[early_stopping]) #menambahkan early
stopping ke dalam callback untuk pelatihan
```

```
#membuat range untuk epoch berdasarkan panjang data loss dari
pelatihan
ephocs_range = range(1, len(history.history['loss']) + 1)

plt.figure(figsize=(10, 10)) #membuat figure dengan ukuran 10x10
untuk menampilkan 2 grafik (Training and Validation Accuracy dan
Loss)

#grafik pertama (Training and Validation Accuracy)
plt.subplot(1, 2, 1) #membuat subplot pertama dalam layout 1 baris
dan 2 kolom
plt.plot(ephocs_range, history.history['accuracy'], label='Training
Accuracy') #plot akurasi pelatihan
plt.plot(ephocs_range, history.history['val_accuracy'],
label='Validation Accuracy') #plot akurasi validasi
plt.legend(loc='lower right') #membuat legenda (informasi elemen
visual) di sudut kanan bawah
plt.xlim(0, 13) #mengatur batas nilai pada sumbu x dari epoch 1
sampai 13
plt.title('Training and Validation Accuracy') #memberi judul grafik

#grafik kedua (Training and Validation Loss)
plt.subplot(1, 2, 2)
plt.plot(ephocs_range, history.history['loss'], label='Training
Loss')
plt.plot(ephocs_range, history.history['val_loss'], label='Validation
Loss')
plt.legend(loc='upper right')
plt.xlim(0, 13)
plt.title('Training and Validation Loss')
plt.show()
```

```
#menyimpan model yang telah dilatih
```

```
model.save('model_mobilenet.h5')
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

#memuat model yang sudah dilatih
model = load_model(r'C:\Users\kevst\OneDrive\Documents\Kuliah\Sem 5\Pembelajaran Mesin dan Pembelajaran Mendalam\UAS\model_mobilenet.h5')  # Ganti dengan path model Anda
class_names = ['Aloevera', 'Echeveria', 'Sedum'] #kelas yang ada pada model

#fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        #memuat dan mempersiapkan gambar untuk prediksi
        input_image = tf.keras.utils.load_img(image_path, target_size=(180, 180)) #membuat gambar dari path dan mnegubah ukurannya menjadi 180x180 pixel
        input_image_array = tf.keras.utils.img_to_array(input_image) #mengubah gambar jadi array numpy agar bisa di proses model
        input_image_exp_dim = tf.expand_dims(input_image_array, 0) #menambahkan dimensi batch agar sesuai dengan input model

#dimensi menjadi (1, 180, 180, 3)

        #melakukan prediksi
        predictions = model.predict(input_image_exp_dim) #melakukan prediksi pada gambar yang telah diproses
        result = tf.nn.softmax(predictions[0]) #menghitung hasil prediksi menggunakan softmax untuk mendapatkan probabilitas tiap kelas
        class_idx = np.argmax(result) #menemukan indeks kelas dengan probabilitas tertinggi
        confidence = np.max(result) * 100 #menghitung confidence dalam persentase

        #menampilkan hasil prediksi dan confidence
```

```
        print(f"Prediksi: {class_names[class_idx]}") #menampilkan
nama kelas yang diprediksi
        print(f"Confidence: {confidence:.2f}%") #menampilkan nilai
confidence

        #menyimpan gambar asli tanpa teks
        input_image = Image.open(image_path) #membuka gambar yang ada
di path
        input_image.save(save_path) #menyimpan gambar asli ke dalam
path yang telah ditentukan

        return f"Prediksi: {class_names[class_idx]} dengan confidence
{confidence:.2f}%. Gambar asli disimpan di {save_path}."
    except Exception as e:
        return f"Terjadi kesalahan: {e}"

#contoh penggunaan fungsi
###Terdapat code yang hilang disini! lihat modul untuk menemukanya

result =
classify_images(r'C:\Users\kevst\OneDrive\Documents\Kuliah\Sem
5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\testing\Aloevera\images (92).jpeg',
save_path='aloevera.jpeg')
print(result)
```

```
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

#memuat model yang telah dilatih sebelumnya
mobileNet_model =
load_model(r'C:\Users\kevst\OneDrive\Documents\Kuliah\Sem
5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\model_mobilenet.h5')#gunakan path masing masing ya

#memuat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\kevst\OneDrive\Documents\Kuliah\Sem 5\Pembelajaran
Mesin dan Pembelajaran Mendalam\UAS\testing', #direktori data uji
```

```
    labels='inferred', #label otomatis dari subfolder yang ada
    label_mode='categorical',  #menghasilkan label dalam bentuk
one-hot encoding
    batch_size=10, #ukuran batch untuk pemrosesan
    image_size=(180, 180) #ukuran gambar yang akan diproses
)

#prediksi model
y_pred = mobileNet_model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)  #konversi ke kelas prediksi

#ekstrak label sebenarnya dari test_data dan konversi ke bentuk
indeks kelas
true_labels = [] #menyimpan label asli dalam bentuk indeks
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())  #konversi
one-hot ke indeks kelas
true_labels = tf.convert_to_tensor(true_labels) #mengkonversi list ke
tensor untuk perhitungan

#membuat confusion matrix untuk evaluasi
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

#menghitung akurasi berdasarkan confusion matrix
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)

#mnghitung presisi dan recall dari confusion matrix
precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=1)

#menghitung F1 Score
f1_score = 2 * (precision * recall) / (precision + recall)

#visualisasi Confusion Matrix
plt.figure(figsize=(6, 5)) #mengatur ukuran gambar
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
#annot=True untuk menampilkan angka di dalam setiap sel matriks

#fmt='d' untuk menampilkan bilangan bulat tanpa desimal
```

```
            xticklabels=["Aloevera", "Echeveria", "Sedum"],
yticklabels=["Aloevera", "Echeveria", "Sedum"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Menampilkan hasil
print("Confusion Matrix:\n", conf_mat.numpy())
print("Akurasi:", accuracy.numpy())
print("Presisi:", precision.numpy())
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())
```

Streamlit:

```
import streamlit as st
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image

model = load_model(r"BestModel_mobilenet_Pandas.h5")
class_names = ["Aloevera", "Echeveria", "Sedum"]

def classify_image(image_path):
    try:
        input_image = tf.keras.utils.load_img(image_path,
target_size=(180, 180))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)

        predictions = model.predict(input_image_exp_dim)
        result = tf.nn.softmax(predictions[0])

        class_idx = np.argmax(result)
        confidence_scores = result.numpy()
        return class_names[class_idx], confidence_scores
    except Exception as e:
        return "Error", str(e)
```

```
def custom_progress_bar(confidence, color1, color2, color3):
    percentage1 = confidence[0] * 100
    percentage2 = confidence[1] * 100
    percentage3 = confidence[2] * 100
    progress_html = f"""
    <div style="border: 1px solid #ddd; border-radius:5px; overflow:
hidden; width: 100%; font-size: 14px;">
        <div style="width: {percentage1:.2f}%; background: {color1};
color: white; text-align: center; height: 24px; float: left;">
            {percentage1:.2f}%
        </div>
        <div style="width: {percentage2:.2f}%; background: {color2};
color: white; text-align: center; height: 24px; float: left;">
            {percentage2:.2f}%
        </div>
        <div style="width: {percentage3:.2f}%; background: {color3};
color: white; text-align: center; height: 24px; float: left;">
            {percentage3:.2f}%
        </div>
    </div>
    """
    st.sidebar.markdown(progress_html, unsafe_allow_html=True)

st.title("Prediksi Jenis Succulent (Aloevera, Echeveria, dan Sedum)")

uploaded_files = st.file_uploader("Unggah Gambar (Beberapa
diperbolehkan)", type=["jpg", "png", "jpeg"],
accept_multiple_files=True)

if st.sidebar.button("Prediksi"):
    if uploaded_files:
        st.sidebar.write("### Hasil Prediksi")
        for uploaded_file in uploaded_files:
            with open(uploaded_file.name, "wb") as f:
                f.write(uploaded_file.getbuffer())

            label, confidence = classify_image(uploaded_file.name)

            if label != "Error":
                primary_color = "#007BFF"
                secondary_color = "#FF4136"
```

```
                tertiary_color = "#2ECC40"
                label_color = primary_color if label == "Aloevera"
else (secondary_color if label == "Echeveria" else tertiary_color)

                st.sidebar.write(f"**Nama File:**
{uploaded_file.name}")
                st.sidebar.markdown(f"<h4 style='color:
{label_color};'>Prediksi: {label}</h4>", unsafe_allow_html=True)

                st.sidebar.write("**Confidence**")
                for i, class_name in enumerate(class_names):
                    st.sidebar.write(f"- {class_name}: {confidence[i]
* 100:.2f}%")

                custom_progress_bar(confidence, primary_color,
secondary_color, tertiary_color)

                st.sidebar.write("---")
            else:
                st.sidebar.error(f"Kesalahan saat memproses gambar
{uploaded_file.name}: {confidence}")
    else:
        st.sidebar.error("Silahkan unggah setidaknya satu gambar
untuk diprediksi.")

if uploaded_files:
    st.write("### Preview Gambar")
    for uploaded_file in uploaded_files:
        image = Image.open(uploaded_file)
        st.image(image, caption=f"{uploaded_file.name}",
use_column_width=True)
```