```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models, Sequential
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

count = 0

dataset_path = r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata'

dirs = os.listdir(dataset_path)
for dir in dirs:
  dir_path = os.path.join(dataset_path, dir)
  if os.path.isdir(dir_path):
    files = os.listdir(dir_path)
    print(f"{dir} Folder has {len(files)} Images")
    count += len(files)

print(f"Total Images in Dataset: {count}")
base_dir = r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata'
img_size = 180
batch_size = 32
validation_split = 0.1
train_datagen = ImageDataGenerator(
```

```
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)



validation_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

print(f'Train samples: {train_generator.samples}')
print(f'Validation samples: {validation_generator.samples}')
print(f'Classes: {train_generator.class_indices}')
class_indices = train_generator.class_indices
```

```
class_names = list(class_indices.keys())

print("Nama-nama kelas:", class_names)

total_count = len(train_generator.filepaths)
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)
num_classes = len(train_generator.class_indices)
print("Number of classes:", num_classes)

def generator_to_dataset(generator, num_classes):
  output_signature = (
    tf.TensorSpec(shape=(None, img_size, img_size, 3), dtype=tf.float32),
    tf.TensorSpec(shape=(None, num_classes), dtype=tf.float32)
  )

  return tf.data.Dataset.from_generator(
    lambda: generator,
    output_signature=output_signature
  )

train_ds = generator_to_dataset(train_generator, num_classes)
val_ds = generator_to_dataset(validation_generator, num_classes)
i = 0
plt.figure(figsize=(10,10))
```

```
images, labels = next(train_generator)
for i in range(min(batch_size, 9)):
    plt.subplot(3, 3, i+1)
    plt.imshow(images[i])
    plt.title(class_names[tf.argmax(labels[i])])
    plt.axis('off')

plt.show()
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)
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape=(img_size, img_size, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory

train_data_dir = r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan
mendalam\Bagian 6\Tugas6_B_11692\Traindata'

train_ds = image_dataset_from_directory(
```

```
    train_data_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(227, 227),
    batch_size=32
)

for images, labels in train_ds.take(1):
    plt.figure(figsize=(10, 10))
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.axis("off")
    plt.show()
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.models import Sequential

# Bangun AlexNet
def build_alexnet(input_shape, num_classes):
    model = Sequential([
        # Input layer
        layers.InputLayer(input_shape=input_shape),

        # Convolutional layers
        layers.Conv2D(96, (11, 11), strides=(4, 4), activation='relu', padding='valid'),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid'),
```

```
        layers.Conv2D(256, (5, 5), activation='relu', padding='same'),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid'),

        layers.Conv2D(384, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(384, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(256, (3, 3), activation='relu', padding='same'),

        # Pooling layer
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid'),

        # Flatten and Dense layers
        layers.Flatten(),
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model

# Misalnya, ukuran gambar input (224x224x3) dan jumlah kelas yang diinginkan
img_size = 224
num_classes = len(class_names)  # Sesuaikan dengan jumlah kelas yang ada

model = build_alexnet(input_shape=(img_size, img_size, 3), num_classes=num_classes)

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Menampilkan struktur model
```

```
model.summary()

model.save('alexnet_model.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 AlexNet yang sudah dilatih

model = load_model(r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\alexnet_model.h5')  # Ganti dengan path model Anda

class_names = ['biji', 'tunas', 'tumbuhan_dewasa'] # 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=(224, 224)) # Mengubah ukuran gambar menjadi 224x224 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, 224, 224, 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}"

result = classify_images(r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\test_data\tumbuhan_dewasa\istockphoto-1322623724-1024x1024.jpg', save_path='matang.jpg')
print(result)

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

# Memuat model AlexNet yang telah dilatih sebelumnya
alexNet_model = load_model(r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\alexnet_model.h5')

# Memuat data uji yang sebenarnya
```

```
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',  # Direktori data uji
    labels='inferred',  # Label otomatis dari subfolder yang ada
    label_mode='categorical',  # Menghasilkan label dalam bentuk one-hot encoding
    batch_size=32,  # Ukuran batch untuk pemrosesan
    image_size=(224, 224)  # Ukuran gambar yang akan diproses sesuai dengan input AlexNet
)

# Definisikan class_names untuk confusion matrix
class_names = test_data.class_names

# Prediksi model
y_pred = alexNet_model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1, output_type=tf.int32)  # Konversi ke kelas prediksi dengan tipe data int32

# Ekstrak label sebenarnya dari test_data dan konversi ke bentuk indeks kelas
true_labels = []  # Menyimpan label asli dalam bentuk indeks
for images, 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, dtype=tf.int32)  # Mengkonversi list ke tensor untuk perhitungan dengan tipe data int32

# Membuat confusion matrix untuk evaluasi
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)
conf_mat = tf.cast(conf_mat, tf.float32)  # Ubah tipe data confusion matrix menjadi float32

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

```
# Menghitung presisi dan recall dari confusion matrix dengan pengecekan untuk menghindari pembagian dengan nol
precision = tf.linalg.diag_part(conf_mat) / (tf.reduce_sum(conf_mat, axis=0) + tf.keras.backend.epsilon())
recall = tf.linalg.diag_part(conf_mat) / (tf.reduce_sum(conf_mat, axis=1) + tf.keras.backend.epsilon())

# Menambahkan debug untuk melihat nilai precision dan recall
print("Precision:", precision.numpy())
print("Recall:", recall.numpy())

# Menghitung F1 Score
f1_score = 2 * (precision * recall) / (precision + recall + tf.keras.backend.epsilon())

# Pastikan F1 score tidak mengandung nilai NaN atau inf (bisa terjadi jika precision dan recall sangat kecil atau 0)
f1_score = tf.where(tf.math.is_nan(f1_score), tf.zeros_like(f1_score), f1_score)

# Visualisasi Confusion Matrix
plt.figure(figsize=(10, 8))  # Mengatur ukuran gambar
sns.heatmap(conf_mat.numpy().astype(int), 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=class_names, yticklabels=class_names)
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())



import tensorflow as tf
import numpy as np
from matplotlib import pyplot as plt
#load data
data_dir = r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata'

data = tf.keras.utils.image_dataset_from_directory(data_dir,seed=123,image_size=(180, 100),batch_size=16)

class_names = data.class_names
print(class_names)

img_size = 180
batch = 32
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)
# Menampilkan beberapa gambar dari dataset untuk melihat bagaimana data terlihat
plt.figure(figsize=(10, 10))
for images, labels in data.take(1):  # Mengambil satu batch pertama
  for i in range(9):  # Menampilkan 9 gambar pertama
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i].numpy()])
    plt.axis("off")

import tensorflow as tf
import numpy as np

# Memuat data dari direktori
data_dir = r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan
mendalam\Bagian 6\Tugas6_B_11692\Traindata'

# Memuat dataset dan resize gambar menjadi 180x180
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,  # Direktori tempat data gambar disimpan
  image_size=(180, 180),  # Ukuran gambar yang diresize
  batch_size=32,  # Ukuran batch untuk pemrosesan
  shuffle=True,  # Mengacak urutan data
```

```
)

# Mengecek bentuk array gambar
for images, labels in train_ds.take(1):
    print("Images shape:", images.shape)
    print("Labels shape:", labels.shape)




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')
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
###Terdapat code yang hilang disini! lihat modul untuk menemukanya

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

```
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, strides=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

# Pastikan input shape dan jumlah kelas sesuai
input_shape = (180, 180, 3)
n_classes = 2

# Clear Cache Keras menggunakan clear session
K.clear_session()

# Buat model dengan
model = googlenet(input_shape, n_classes)
model.summary()
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

# Compile model dengan optimizer Adam
model.compile(
    optimizer=Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Buat early stopping
```

```
early_stopping = EarlyStopping(
    monitor='val_accuracy',
    patience=5,
    mode='max'
)
if 'history' in globals():
    epochs_range = range(1, len(history.history['loss']) + 1)
    plt.figure(figsize=(10, 10))

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

    plt.show()
else:
    print("History belum didefinisikan. Pastikan model sudah dilatih terlebih dahulu.")

model.save('gugelnet.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'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\gugelnet.h5')  # Ganti dengan path model Anda
class_names = ['tunas', 'tumbuhan_dewasa', 'biji']

# 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=(180, 180))
    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'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\test_data\tunas\istockphoto-517610218-612x612.jpg', save_path='mentah2.jpg')
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

# Muat model yang telah dilatih
model_path = r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\gugelnet.h5'  # Ganti dengan path ke file model Anda
model = load_model(model_path)

# Muat data uji
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',  # Ganti dengan path ke folder data uji Anda
    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=["Mentah", "Matang"], yticklabels=["Mentah", "Matang"])
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())



import os
import numpy as np
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, Flatten, Dense, Dropout
count = 0
dirs = os.listdir(r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata')
for dir in dirs:
   files = list(os.listdir(r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata/'+dir))
   print(dir + ' Folder has ' + str(len(files)) + ' Images ')
   count = count + len(files)
print('Images Folder has ' + str(count) + 'Images')
base_dir = r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata'
img_size = 224
batch = 32
```

```
validation_split = 0.1
test_split = 0.1
import tensorflow as tf
dataset = tf.keras.utils.image_dataset_from_directory(
  base_dir,
  seed=123,
  image_size=(img_size, img_size),
  batch_size=batch,
)
class_names = dataset.class_names
print("Class Names:", class_names)
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))

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')
import numpy as np

for images, labels in train_ds.take(1):
   images_array = np.array(images)
   print(images_array.shape)
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.layers import Rescaling
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras import Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers

data_augmentation = Sequential([
   layers.RandomFlip("horizontal_and_vertical"),
   layers.RandomRotation(0.2),
   layers.RandomZoom(0.2),
])

base_model = MobileNet(weights=None, include_top=False, input_shape=(224, 224, 3))
base_model.trainable = True

fine_tune_at = len(base_model.layers)
```

```
for layer in base_model.layers[:fine_tune_at]:
  layer.trainable = False

model =  Sequential([Input(shape=(224, 224, 3)),
            data_augmentation, layers.Rescaling(1./255), base_model,
            layers.GlobalAveragePooling2D(), Dense(128, activation='relu'),
            Dropout(0.3), Dense(3, activation='softmax')])
from tensorflow.keras.optimizers import Adam

model.compile(
  optimizer=Adam(learning_rate=1e-4),
  loss='sparse_categorical_crossentropy',
  metrics=['accuracy']
)
model.summary()
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                    patience=3,
                    mode='max')
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=30,
  callbacks=[early_stopping]
)
epoch_range = range(1, len(history.history['loss']) + 1)

plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
```

```
plt.plot(epoch_range, history.history['accuracy'], label='Training Accuracy')
if 'val_accuracy' in history.history:
  plt.plot(epoch_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(epoch_range, history.history['loss'], label='Training Loss')
if 'val_loss' in history.history:
  plt.plot(epoch_range, history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
model.save('BestModel_MobileNet_Numpy.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
def classify_images(image_path, save_path='datatest.jpg'):
  try:
    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)

    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}"

image_paths = [
    r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\test_data\tunas\istockphoto-517610218-612x612.jpg',
    r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\test_data\tumbuhan_dewasa\istockphoto-1322623724-1024x1024.jpg',
]

# Loop melalui semua gambar
for i, image_path in enumerate(image_paths, start=1):
    save_path = f"datatest_{i}.jpg"
    result = classify_images(image_path, save_path=save_path)
    print(result)
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
import seaborn as sns
mobileNet_model = load_model(r'BestModel_MobileNet_Numpy.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\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\BestModel_MobileNet_Numpy.h5')  # Ganti dengan path model Anda

class_names = ['tunas', 'benih', 'tumbuhan_dewasa'] #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\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\test_data\tunas\istockphoto-517610218-612x612.jpg', save_path='tunas.jpg')
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\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\BestModel_MobileNet_Numpy.h5')#gunakan path masing masing ya

#memuat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
```

```
    r'test_data', #direktori data uji
    labels='inferred', #label otomatis dari subfolder yang ada
    label_mode='categorical',  #menghasilkan label dalam bentuk one-hot encoding
    batch_size=32, #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=["tunas", "biji", "tumbuhan_dewasa"], yticklabels=["tunas", "biji", "tumbuhan_dewasa"])

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())



#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\LENOVO\Documents\Semester 5\Pembelajaran mesin dan
mendalam\Bagian 6\Tugas6_B_11692\Traindata')

for dir in dirs:

   files = list(os.listdir(r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan
mendalam\Bagian 6\Tugas6_B_11692\Traindata/'+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\LENOVO\Documents\Semester 5\Pembelajaran mesin dan
mendalam\Bagian 6\Tugas6_B_11692\Traindata' #direktori folder dataset

img_size = 180 #mengubah ukuran gambar menjadi 180

batch = 32 #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)
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

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("harizontal", 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 VGG16 #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 = VGG16(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


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

early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=4,
                               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
import matplotlib.pyplot as plt

# Mengatur rentang epoch
epochs_range = range(1, len(history.history['loss']) + 1)

# Membuat plot
plt.figure(figsize=(10, 10))

# Plot Akurasi
plt.subplot(1, 2, 1)
plt.plot(epochs_range, history.history['accuracy'], label='Training Accuracy')
if 'val_accuracy' in history.history:
   plt.plot(epochs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

# Plot Loss
plt.subplot(1, 2, 2)
plt.plot(epochs_range, history.history['loss'], label='Training Loss')
if 'val_loss' in history.history:
   plt.plot(epochs_range, history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
```

```
# Menampilkan plot
plt.show()
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import os

# 1. Load VGG16 Pretrained Model
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
for layer in base_model.layers:
    layer.trainable = False
model = Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')  # 10 classes (modify based on your dataset)
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# 5. Data Preparation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
```

```
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata',  # Update with your dataset path
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\Traindata',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
model.save('models/vgg16_final_model.h5')
callbacks = [
    EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True),
    ModelCheckpoint('models/vgg16_best_model.keras', save_best_only=True)  # Ubah ke .keras
]
```

```
# Simpan model akhir dengan .keras

model.save('models/vgg16_final_model.keras')

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\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\models\vgg16_final_model.h5')  # Ganti dengan path model Anda

class_names = ['Matang', 'Mentah'] #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'D:\Semester 5\Pembelajaran Mesin Dan Mendalam\UAS\train_data\Dewasa_strawberry\cara-menanam-strawberry-dari-biji.jpg', save_path='matang.jpg')
print(result)

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

# Memuat model VGG16 yang telah dilatih sebelumnya
```

```
vgg16_model = load_model(r'C:\Users\LENOVO\Documents\Semester 5\Pembelajaran mesin dan mendalam\Bagian 6\Tugas6_B_11692\models\vgg16_final_model.h5')  # Ganti dengan path sesuai sistem Anda


# Memuat data uji
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',  # Ganti dengan path ke direktori data uji
    labels='inferred',  # Label otomatis dari subfolder
    label_mode='categorical',  # Menghasilkan label dalam bentuk one-hot encoding
    batch_size=32,  # Ukuran batch untuk pemrosesan
    image_size=(224, 224)  # Ukuran gambar yang sesuai untuk VGG16
)

# Prediksi menggunakan model
y_pred = vgg16_model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)  # Konversi prediksi menjadi kelas

# Ekstrak label sebenarnya dari test_data dan konversi ke 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)  # Mengubah list ke tensor

# Membuat confusion matrix
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 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=(8, 6))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
        xticklabels=test_data.class_names,  # Label prediksi
        yticklabels=test_data.class_names)  # Label asli
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())
```