```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import load_img,
ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image

base_dir = r'C:\Users\ROG STRIX\Documents\000000000000000 MESIN\Projek
UAS PMDPM_A_Pandas\train_data'
test_data_dir = r'C:\Users\ROG STRIX\Documents\000000000000000 MESIN\
Projek UAS PMDPM_A_Pandas\test_data'

img_size = 180
batch_size = 32
validation_split = 0.1
test_split = 0.1

dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch_size,
    validation_split=validation_split + test_split,  # 20% untuk
validasi dan test
    subset="training",
    interpolation="bilinear"
)

# Pembagian lebih lanjut untuk validasi dan test (sisa dari 20%)
val_dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch_size,
    validation_split=validation_split + test_split,
    subset="validation",
    interpolation="bilinear"
)

Found 300 files belonging to 3 classes.
Using 240 files for training.
Found 300 files belonging to 3 classes.
Using 60 files for validation.
```

```
class_names = dataset.class_names
print("Class Names:", class_names)
```

```
Class Names: ['Jeruk Lemon', 'Jeruk Nipis', 'Jeruk Sunkist']
```

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds =
dataset.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds =
val_dataset.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)
])
```

```
d:\anaconda3\Lib\site-packages\keras\src\layers\preprocessing\
tf_data_layer.py:19: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
def build_alexnet(input_shape=(img_size, img_size, 3)):
    model = models.Sequential([
        # Layer 1: Convolutional Layer 1
        layers.Conv2D(96, (11, 11), strides=(4, 4), activation='relu',
input_shape=input_shape),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),

        # Layer 2: Convolutional Layer 2
        layers.Conv2D(256, (5, 5), padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),

        # Layer 3: Convolutional Layer 3
        layers.Conv2D(384, (3, 3), padding='same', activation='relu'),

        # Layer 4: Convolutional Layer 4
        layers.Conv2D(384, (3, 3), padding='same', activation='relu'),

        # Layer 5: Convolutional Layer 5
        layers.Conv2D(256, (3, 3), padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)),

        # Flatten the output of convolutional layers
        layers.Flatten(),

        # Fully Connected Layer 1
```

```
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),

        # Fully Connected Layer 2
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),

        # Output Layer
        layers.Dense(len(class_names), activation='softmax')  # 
`len(class_names)` disesuaikan dengan jumlah kelas
    ])

    return model

# Bangun model AlexNet tanpa pre-trained weights
model_alexnet = build_alexnet()

# Kompilasi model
model_alexnet.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

```
d:\anaconda3\Lib\site-packages\keras\src\layers\convolutional\
base_conv.py:107: UserWarning: Do not pass an 
`input_shape`/`input_dim` argument to a layer. When using Sequential 
models, prefer using an `Input(shape)` object as the first layer in 
the model instead.
  super().__init__(activity_regularizer=activity_regularizer, 
**kwargs)
```

```
model_alexnet.summary()
```

```
Model: "sequential_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃ 
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 43, 43, 96)          │ 
34,944 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 21, 21, 96)          │ 
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
```

```
| conv2d_1 (Conv2D)                | (None, 21, 21, 256)     |
614,656 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| max_pooling2d_1 (MaxPooling2D)   | (None, 10, 10, 256)     |
0 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_2 (Conv2D)                | (None, 10, 10, 384)     |
885,120 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_3 (Conv2D)                | (None, 10, 10, 384)     |
1,327,488 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_4 (Conv2D)                | (None, 10, 10, 256)     |
884,992 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| max_pooling2d_2 (MaxPooling2D)   | (None, 4, 4, 256)       |
0 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| flatten (Flatten)                | (None, 4096)            |
0 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| dense (Dense)                    | (None, 4096)            |
16,781,312 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| dropout (Dropout)                | (None, 4096)            |
0 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| dense_1 (Dense)                  | (None, 4096)            |
16,781,312 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| dropout_1 (Dropout)              | (None, 4096)            |
0 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| dense_2 (Dense)                  | (None, 3)               |
12,291 |
└──────────────────────────────────┴─────────────────────────┴──────────
──────┘

 Total params: 37,322,115 (142.37 MB)
```

```
 Trainable params: 37,322,115 (142.37 MB)

 Non-trainable params: 0 (0.00 B)
```

```
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3,
mode='max')

history = model_alexnet.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=[early_stopping]
)
```

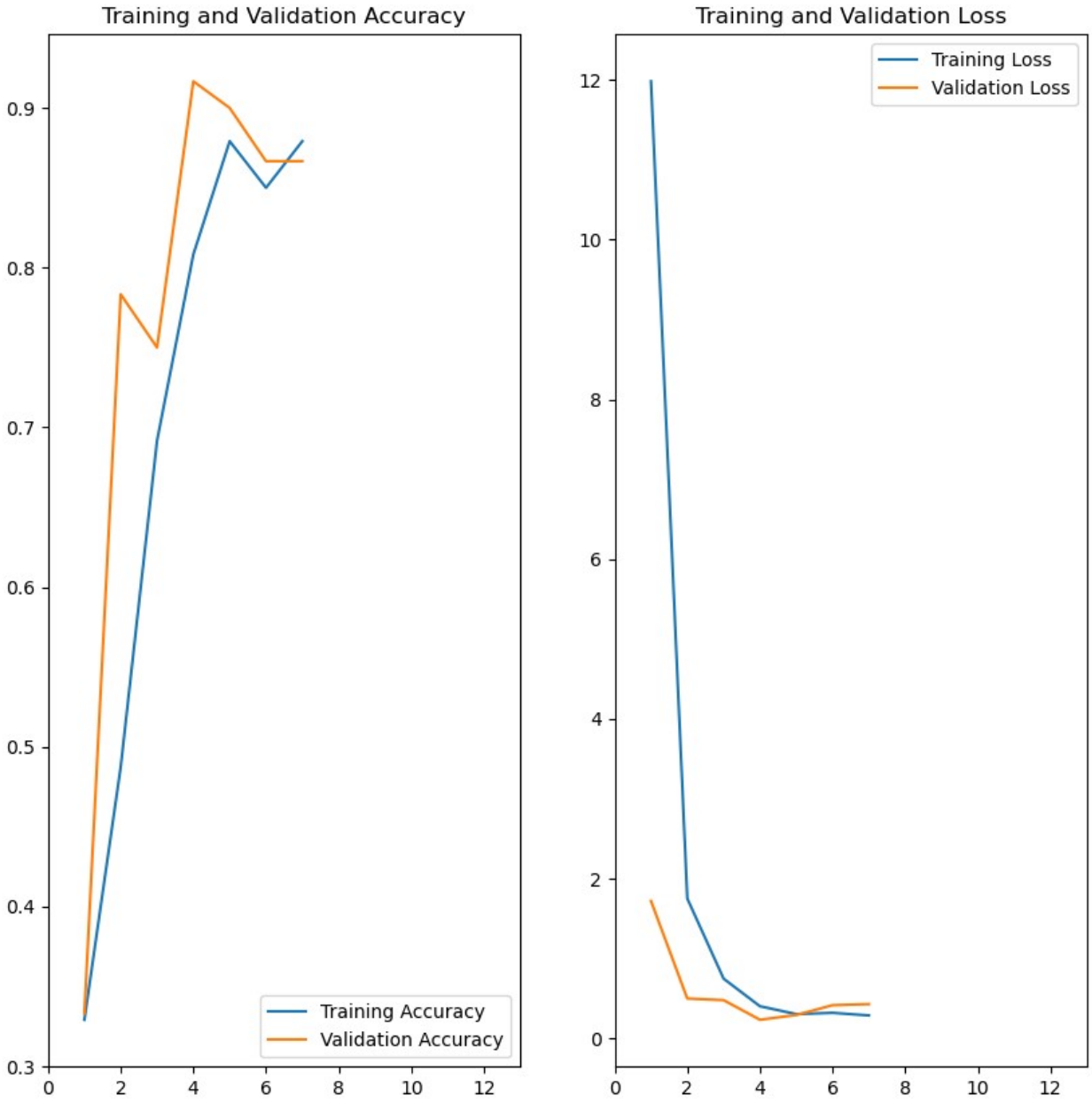
```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 6s 401ms/step - accuracy: 0.3219 - loss:
15.6481 - val_accuracy: 0.3333 - val_loss: 1.7217
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 349ms/step - accuracy: 0.4872 - loss:
2.0299 - val_accuracy: 0.7833 - val_loss: 0.5000
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 346ms/step - accuracy: 0.6732 - loss:
0.8330 - val_accuracy: 0.7500 - val_loss: 0.4797
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 344ms/step - accuracy: 0.7683 - loss:
0.4369 - val_accuracy: 0.9167 - val_loss: 0.2344
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 342ms/step - accuracy: 0.8804 - loss:
0.3230 - val_accuracy: 0.9000 - val_loss: 0.2936
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 335ms/step - accuracy: 0.8542 - loss:
0.3062 - val_accuracy: 0.8667 - val_loss: 0.4165
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 336ms/step - accuracy: 0.8826 - loss:
0.2660 - val_accuracy: 0.8667 - val_loss: 0.4297
```

```
model_alexnet.save('BestModel_AlexNet_Pandas.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

```
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.xlim(0, 13)
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.xlim(0, 13)
plt.title('Training and Validation Loss')
plt.show()
```




```
def classify_images(image_path, save_path='predicted_image.jpg'):
    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_alexnet.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}"

# Uji prediksi pada gambar baru
result = classify_images(r'test_data/Jeruk
Sunkist/jeruk_sunkist_test_3.jpeg', save_path='sunkist_2.jpg')
print(result)
```

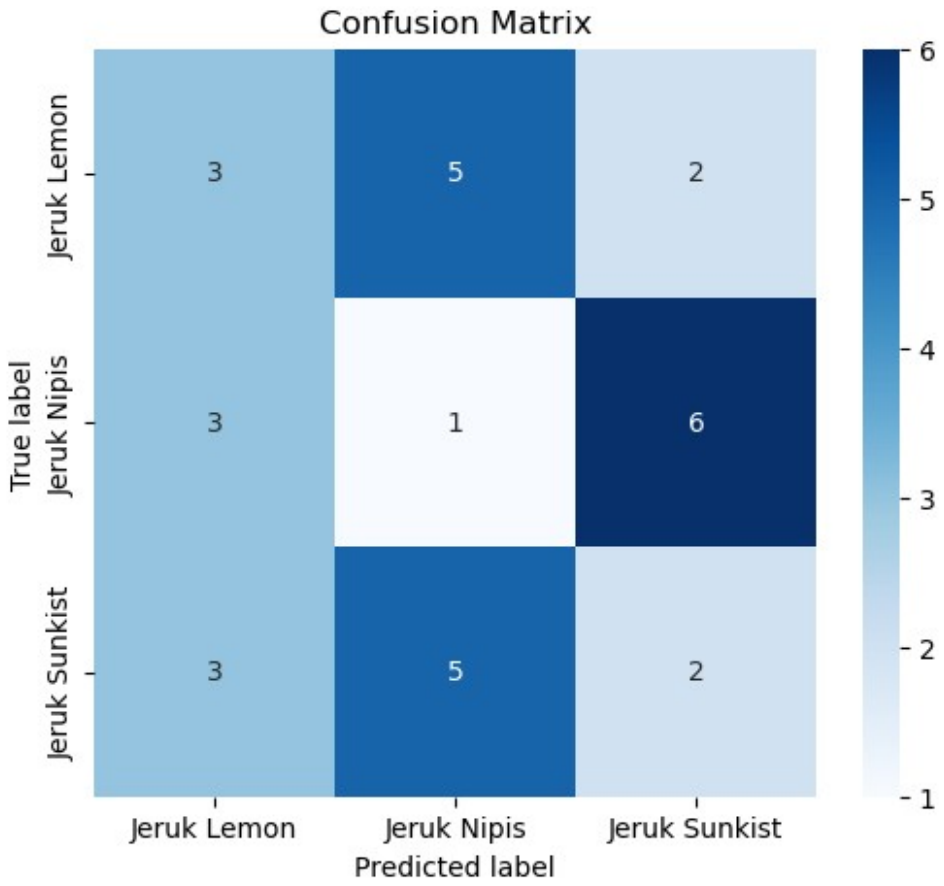
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 105ms/step
Prediksi: Jeruk Sunkist
Confidence: 56.83%
Prediksi: Jeruk Sunkist dengan confidence 56.83%. Gambar asli disimpan
di sunkist_2.jpg.
```

```
from tensorflow.keras.models import load_model

# Muat model yang telah dilatih
model_alexnet = load_model('BestModel_AlexNet_Pandas.h5')

# Persiapkan dataset pengujian
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    test_data_dir,
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

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

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

# Matriks kebingungannya
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)
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=["Jeruk Lemon", "Jeruk Nipis", "Jeruk
Sunkist"], yticklabels=["Jeruk Lemon", "Jeruk Nipis", "Jeruk
Sunkist"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Tampilkan hasil evaluasi
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())
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

```
Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 197ms/step
```




```
Confusion Matrix:
 [[3 5 2]
 [3 1 6]
 [3 5 2]]
Akurasi: 0.2
Presisi: [0.33333333 0.09090909 0.2       ]
Recall: [0.3 0.1 0.2]
F1 Score: [0.31578947 0.0952381  0.2       ]
```

```
import tensorflow as tf
import cv2
import numpy as np
from matplotlib import pyplot as plt

data_dir = r"C:\Users\ROG STRIX\Documents\000000000000000 MESIN\Projek
UAS PMDPM_A_Pandas\train_data"

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

print(data.class_names)

class_names = data.class_names
```

```
Found 300 files belonging to 3 classes.
['Jeruk Lemon', 'Jeruk Nipis', 'Jeruk Sunkist']
```

```
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,
)
```

```
Found 300 files belonging to 3 classes.
```

```
total_count = len(dataset)
train_count = int(0.8 * total_count)
val_count = int(0.1 * total_count)
test_count = total_count - train_count - val_count

train_ds = dataset.take(train_count)
remaining_ds = dataset.skip(train_count)
val_ds = remaining_ds.take(val_count)
test_ds = remaining_ds.skip(val_count)

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

```
Total Images: 10
Train Images: 8
Validation Images: 1
Test Images: 1
```

```
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')
```

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

```
(32, 180, 180, 3)
```

```
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)
test_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')
```

```
d:\anaconda3\Lib\site-packages\keras\src\layers\preprocessing\
tf_data_layer.py:19: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
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 = 180, 180, 3
n_classes = 3

K.clear_session()

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

```
WARNING:tensorflow:From d:\anaconda3\Lib\site-packages\keras\src\
backend\common\global_state.py:82: The name tf.reset_default_graph is
deprecated. Please use tf.compat.v1.reset_default_graph instead.


Model: "functional"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 180, 180, 3)	0	-
conv2d (Conv2D)	(None, 90, 90, 64)	9,472	input_layer[0][0]
max_pooling2d (MaxPooling2D)	(None, 45, 45, 64)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 45, 45, 64)	4,160	max_pooling2d[0]…
conv2d_2 (Conv2D)	(None, 45, 45, 192)	110,784	conv2d_1[0][0]

```
──────┤
| max_pooling2d_1     | (None, 22, 22,    |           0 | conv2d_2[0]
[0]    |
| (MaxPooling2D)      | 192)              |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
| conv2d_4 (Conv2D)   | (None, 22, 22,    |      18,528 |
max_pooling2d_1[… |
|                     | 96)               |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
| conv2d_6 (Conv2D)   | (None, 22, 22,    |       3,088 |
max_pooling2d_1[… |
|                     | 16)               |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
| max_pooling2d_2     | (None, 22, 22,    |           0 |
max_pooling2d_1[… |
| (MaxPooling2D)      | 192)              |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
| conv2d_3 (Conv2D)   | (None, 22, 22,    |      12,352 |
max_pooling2d_1[… |
|                     | 64)               |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
| conv2d_5 (Conv2D)   | (None, 22, 22,    |     110,720 | conv2d_4[0]
[0]    |
|                     | 128)              |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
| conv2d_7 (Conv2D)   | (None, 22, 22,    |      12,832 | conv2d_6[0]
[0]    |
|                     | 32)               |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
| conv2d_8 (Conv2D)   | (None, 22, 22,    |       6,176 |
max_pooling2d_2[… |
|                     | 32)               |             |
|
├─────────────────────┼───────────────────┼─────────────┼──────────────
──────┤
```

concatenate (Concatenate)	(None, 22, 22, 256)	0	conv2d_3[0][0], conv2d_5[0][0], conv2d_7[0][0], conv2d_8[0][0]
conv2d_10 (Conv2D)	(None, 22, 22, 128)	32,896	concatenate[0][0]
conv2d_12 (Conv2D)	(None, 22, 22, 32)	8,224	concatenate[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 22, 22, 256)	0	concatenate[0][0]
conv2d_9 (Conv2D)	(None, 22, 22, 128)	32,896	concatenate[0][0]
conv2d_11 (Conv2D)	(None, 22, 22, 192)	221,376	conv2d_10[0][0]
conv2d_13 (Conv2D)	(None, 22, 22, 96)	76,896	conv2d_12[0][0]
conv2d_14 (Conv2D)	(None, 22, 22, 64)	16,448	max_pooling2d_3[…

```
|
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
| concatenate_1        | (None, 22, 22,       |           0 | conv2d_9[0]
[0],   |
| (Concatenate)        | 480)                 |             | conv2d_11[0]
[0],  |
|                      |                      |             | conv2d_13[0]
[0],  |
|                      |                      |             | conv2d_14[0]
[0]   |
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
| max_pooling2d_4      | (None, 11, 11,       |           0 |
concatenate_1[0]… |
| (MaxPooling2D)       | 480)                 |             |
|
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
| conv2d_16 (Conv2D)   | (None, 11, 11,       |      46,176 |
max_pooling2d_4[… |
|                      | 96)                  |             |
|
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
| conv2d_18 (Conv2D)   | (None, 11, 11,       |       7,696 |
max_pooling2d_4[… |
|                      | 16)                  |             |
|
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
| max_pooling2d_5      | (None, 11, 11,       |           0 |
max_pooling2d_4[… |
| (MaxPooling2D)       | 480)                 |             |
|
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
| conv2d_15 (Conv2D)   | (None, 11, 11,       |      92,352 |
max_pooling2d_4[… |
|                      | 192)                 |             |
|
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
| conv2d_17 (Conv2D)   | (None, 11, 11,       |     179,920 | conv2d_16[0]
[0]   |
|                      | 208)                 |             |
|
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
```

conv2d_19 (Conv2D)	(None, 11, 11, 48)	19,248	conv2d_18[0][0]
conv2d_20 (Conv2D)	(None, 11, 11, 64)	30,784	max_pooling2d_5[…
concatenate_2 (Concatenate)	(None, 11, 11, 512)	0	conv2d_15[0][0], conv2d_17[0][0], conv2d_19[0][0], conv2d_20[0][0]
conv2d_22 (Conv2D)	(None, 11, 11, 112)	57,456	concatenate_2[0]…
conv2d_24 (Conv2D)	(None, 11, 11, 24)	12,312	concatenate_2[0]…
max_pooling2d_6 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_2[0]…
conv2d_21 (Conv2D)	(None, 11, 11, 160)	82,080	concatenate_2[0]…
conv2d_23 (Conv2D)	(None, 11, 11, 224)	226,016	conv2d_22[0][0]

```
│
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
│ conv2d_25 (Conv2D)   │ (None, 11, 11,       │     38,464 │ conv2d_24[0]
[0]   │
│                      │ 64)                  │            │
│
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
│ conv2d_26 (Conv2D)   │ (None, 11, 11,       │     32,832 │
max_pooling2d_6[… │
│                      │ 64)                  │            │
│
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
│ concatenate_3        │ (None, 11, 11,       │          0 │ conv2d_21[0]
[0],  │
│ (Concatenate)        │ 512)                 │            │ conv2d_23[0]
[0],  │
│                      │                      │            │ conv2d_25[0]
[0],  │
│                      │                      │            │ conv2d_26[0]
[0]   │
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
│ conv2d_28 (Conv2D)   │ (None, 11, 11,       │     65,664 │
concatenate_3[0][… │
│                      │ 128)                 │            │
│
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
│ conv2d_30 (Conv2D)   │ (None, 11, 11,       │     12,312 │
concatenate_3[0][… │
│                      │ 24)                  │            │
│
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
│ max_pooling2d_7      │ (None, 11, 11,       │          0 │
concatenate_3[0][… │
│ (MaxPooling2D)       │ 512)                 │            │
│
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
│ conv2d_27 (Conv2D)   │ (None, 11, 11,       │     65,664 │
concatenate_3[0][… │
│                      │ 128)                 │            │
│
├──────────────────────┼──────────────────────┼──────────────┼──────────────────
──────┤
```

conv2d_29 (Conv2D)	(None, 11, 11, 256)	295,168	conv2d_28[0][0]
conv2d_31 (Conv2D)	(None, 11, 11, 64)	38,464	conv2d_30[0][0]
conv2d_32 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_7[…
concatenate_4 (Concatenate)	(None, 11, 11, 512)	0	conv2d_27[0][0], conv2d_29[0][0], conv2d_31[0][0], conv2d_32[0][0]
conv2d_34 (Conv2D)	(None, 11, 11, 144)	73,872	concatenate_4[0]…
conv2d_36 (Conv2D)	(None, 11, 11, 32)	16,416	concatenate_4[0]…
max_pooling2d_8 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_4[0]…
conv2d_33 (Conv2D)	(None, 11, 11, 112)	57,456	concatenate_4[0]…

```
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_35 (Conv2D)    │ (None, 11, 11,     │     373,536 │ conv2d_34[0]
[0]    │
│                       │ 288)               │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_37 (Conv2D)    │ (None, 11, 11,     │      51,264 │ conv2d_36[0]
[0]    │
│                       │ 64)                │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_38 (Conv2D)    │ (None, 11, 11,     │      32,832 │
max_pooling2d_8[… │
│                       │ 64)                │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ concatenate_5         │ (None, 11, 11,     │           0 │ conv2d_33[0]
[0],   │
│ (Concatenate)         │ 528)               │             │ conv2d_35[0]
[0],   │
│                       │                    │             │ conv2d_37[0]
[0],   │
│                       │                    │             │ conv2d_38[0]
[0]    │
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_40 (Conv2D)    │ (None, 11, 11,     │      84,640 │
concatenate_5[0]… │
│                       │ 160)               │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_42 (Conv2D)    │ (None, 11, 11,     │      16,928 │
concatenate_5[0]… │
│                       │ 32)                │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ max_pooling2d_9       │ (None, 11, 11,     │           0 │
concatenate_5[0]… │
│ (MaxPooling2D)        │ 528)               │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
```

```
| conv2d_39 (Conv2D)  | (None, 11, 11,     |    135,424 |
concatenate_5[0][0… |
|                     | 256)               |            |
|
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| conv2d_41 (Conv2D)  | (None, 11, 11,     |    461,120 | conv2d_40[0]
[0]   |
|                     | 320)               |            |
|
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| conv2d_43 (Conv2D)  | (None, 11, 11,     |    102,528 | conv2d_42[0]
[0]   |
|                     | 128)               |            |
|
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| conv2d_44 (Conv2D)  | (None, 11, 11,     |     67,712 |
max_pooling2d_9[0][… |
|                     | 128)               |            |
|
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| concatenate_6       | (None, 11, 11,     |          0 | conv2d_39[0]
[0],  |
| (Concatenate)       | 832)               |            | conv2d_41[0]
[0],  |
|                     |                    |            | conv2d_43[0]
[0],  |
|                     |                    |            | conv2d_44[0]
[0]   |
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| max_pooling2d_10    | (None, 6, 6, 832)  |          0 |
concatenate_6[0][0… |
| (MaxPooling2D)      |                    |            |
|
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| conv2d_46 (Conv2D)  | (None, 6, 6, 160)  |    133,280 |
max_pooling2d_10[0… |
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| conv2d_48 (Conv2D)  | (None, 6, 6, 32)   |     26,656 |
max_pooling2d_10[0… |
├─────────────────────┼────────────────────┼────────────┼──────────────
──────┤
| max_pooling2d_11    | (None, 6, 6, 832)  |          0 |
```

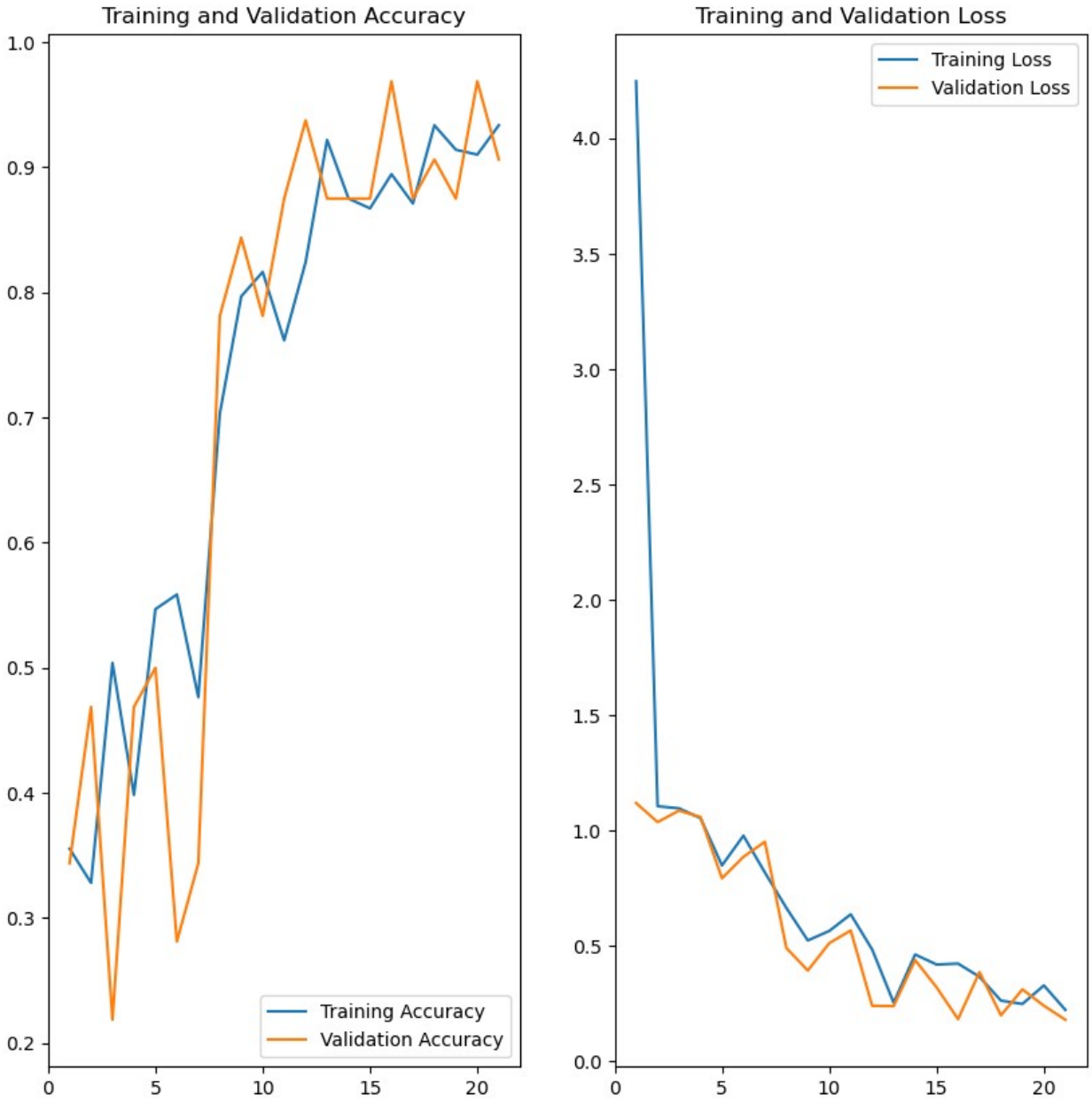
```
max_pooling2d_10… │
│ (MaxPooling2D)      │                   │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_45 (Conv2D)  │ (None, 6, 6, 256) │    213,248 │
max_pooling2d_10… │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_47 (Conv2D)  │ (None, 6, 6, 320) │    461,120 │ conv2d_46[0]
[0]   │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_49 (Conv2D)  │ (None, 6, 6, 128) │    102,528 │ conv2d_48[0]
[0]   │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_50 (Conv2D)  │ (None, 6, 6, 128) │    106,624 │
max_pooling2d_11… │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ concatenate_7       │ (None, 6, 6, 832) │          0 │ conv2d_45[0]
[0],  │
│ (Concatenate)       │                   │            │ conv2d_47[0]
[0],  │
│                     │                   │            │ conv2d_49[0]
[0],  │
│                     │                   │            │ conv2d_50[0]
[0]   │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_52 (Conv2D)  │ (None, 6, 6, 192) │    159,936 │
concatenate_7[0][0… │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_54 (Conv2D)  │ (None, 6, 6, 48)  │     39,984 │
concatenate_7[0][0… │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ max_pooling2d_12    │ (None, 6, 6, 832) │          0 │
concatenate_7[0][0… │
│ (MaxPooling2D)      │                   │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_51 (Conv2D)  │ (None, 6, 6, 384) │    319,872 │
concatenate_7[0][0… │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
```

```
| conv2d_53 (Conv2D)  | (None, 6, 6, 384) |    663,936 | conv2d_52[0]
[0]   |
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
| conv2d_55 (Conv2D)  | (None, 6, 6, 128) |    153,728 | conv2d_54[0]
[0]   |
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
| conv2d_56 (Conv2D)  | (None, 6, 6, 128) |    106,624 |
max_pooling2d_12… |
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
| concatenate_8       | (None, 6, 6,      |          0 | conv2d_51[0]
[0],  |
| (Concatenate)       | 1024)             |            | conv2d_53[0]
[0],  |
|                     |                   |            | conv2d_55[0]
[0],  |
|                     |                   |            | conv2d_56[0]
[0]   |
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
| average_pooling2d   | (None, 4, 4,      |          0 |
concatenate_8[0]… |
| (AveragePooling2D)  | 1024)             |            |
|
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
| dropout (Dropout)   | (None, 4, 4,      |          0 |
average_pooling2… |
|                     | 1024)             |            |
|
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
| flatten (Flatten)   | (None, 16384)     |          0 | dropout[0][0]
|
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
| dense (Dense)       | (None, 3)         |     49,155 | flatten[0][0]
|
└─────────────────────┴───────────────────┴────────────┴──────────────
──────┘

 Total params: 6,022,707 (22.97 MB)

 Trainable params: 6,022,707 (22.97 MB)

 Non-trainable params: 0 (0.00 B)
```

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

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

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

history= model.fit(train_ds,
                   epochs=30,
                   validation_data=val_ds,
                   callbacks=[early_stopping])
```

```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 20s 524ms/step - accuracy: 0.3916 - loss:
6.1813 - val_accuracy: 0.3438 - val_loss: 1.1201
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 395ms/step - accuracy: 0.3367 - loss:
1.1082 - val_accuracy: 0.4688 - val_loss: 1.0376
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 437ms/step - accuracy: 0.4537 - loss:
1.0750 - val_accuracy: 0.2188 - val_loss: 1.0875
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 432ms/step - accuracy: 0.3600 - loss:
1.0684 - val_accuracy: 0.4688 - val_loss: 1.0595
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 441ms/step - accuracy: 0.5062 - loss:
0.9180 - val_accuracy: 0.5000 - val_loss: 0.7942
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 427ms/step - accuracy: 0.6273 - loss:
0.8161 - val_accuracy: 0.2812 - val_loss: 0.8871
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 400ms/step - accuracy: 0.4587 - loss:
0.8186 - val_accuracy: 0.3438 - val_loss: 0.9524
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 401ms/step - accuracy: 0.7194 - loss:
0.6720 - val_accuracy: 0.7812 - val_loss: 0.4921
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 398ms/step - accuracy: 0.7591 - loss:
0.5708 - val_accuracy: 0.8438 - val_loss: 0.3945
Epoch 10/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 391ms/step - accuracy: 0.8360 - loss:
0.4702 - val_accuracy: 0.7812 - val_loss: 0.5130
Epoch 11/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 393ms/step - accuracy: 0.7412 - loss:
```

```
0.6373 - val_accuracy: 0.8750 - val_loss: 0.5681
Epoch 12/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 391ms/step - accuracy: 0.8331 - loss:
0.5424 - val_accuracy: 0.9375 - val_loss: 0.2414
Epoch 13/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 396ms/step - accuracy: 0.9030 - loss:
0.2694 - val_accuracy: 0.8750 - val_loss: 0.2405
Epoch 14/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 388ms/step - accuracy: 0.8833 - loss:
0.4770 - val_accuracy: 0.8750 - val_loss: 0.4398
Epoch 15/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 395ms/step - accuracy: 0.8706 - loss:
0.4271 - val_accuracy: 0.8750 - val_loss: 0.3216
Epoch 16/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 405ms/step - accuracy: 0.9065 - loss:
0.3340 - val_accuracy: 0.9688 - val_loss: 0.1839
Epoch 17/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 385ms/step - accuracy: 0.8798 - loss:
0.3383 - val_accuracy: 0.8750 - val_loss: 0.3878
Epoch 18/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 394ms/step - accuracy: 0.9176 - loss:
0.3313 - val_accuracy: 0.9062 - val_loss: 0.2004
Epoch 19/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 387ms/step - accuracy: 0.9271 - loss:
0.2036 - val_accuracy: 0.8750 - val_loss: 0.3129
Epoch 20/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 397ms/step - accuracy: 0.8741 - loss:
0.4381 - val_accuracy: 0.9688 - val_loss: 0.2419
Epoch 21/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 394ms/step - accuracy: 0.9442 - loss:
0.2205 - val_accuracy: 0.9062 - val_loss: 0.1813


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_GoogleNet_Pandas.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

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

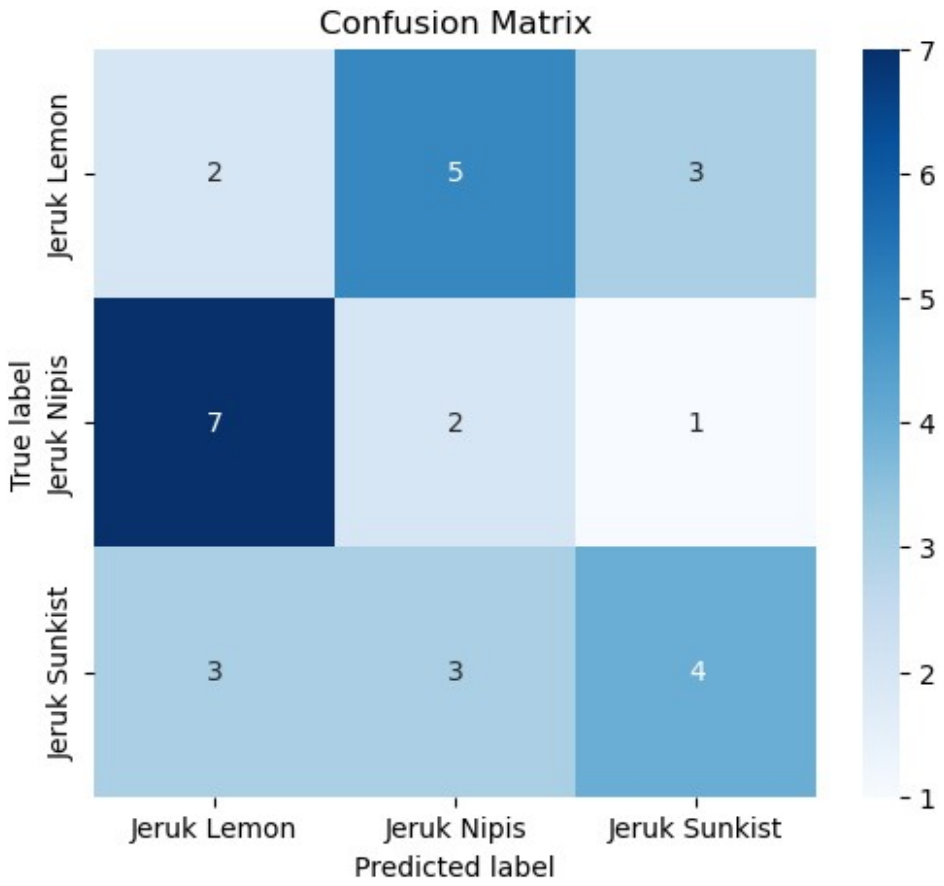
```
from PIL import Image

model = load_model(r'C:\Users\ROG STRIX\Documents\000000000000000
MESIN\Projek UAS PMDPM_A_Pandas\BestModel_GoogleNet_Pandas.h5')
class_names = ['Jeruk Lemon', 'Jeruk Nipis', 'Jeruk Sunkist']

def classify_images(image_path, save_path='predicted_image.jpg'):
    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 = 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'test_data/Jeruk
Nipis/jeruk_nipis_test_4.jpeg', save_path='nipis.jpg')
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 939ms/step
Prediksi: Jeruk Nipis
Confidence: 55.55%
Prediksi: Jeruk Nipis dengan confidence 55.55%. Gambar asli disimpan
di nipis.jpg.
```

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

test_data = tf.keras.preprocessing.image_dataset_from_directory(
```

```
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

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)

f1_score = 2 * (precision * recall) / (precision + recall)

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

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())
```

```
Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 745ms/step
```




```
Confusion Matrix:
 [[2 5 3]
 [7 2 1]
 [3 3 4]]
Akurasi: 0.26666666666666666
Presisi: [0.16666667 0.2        0.5       ]
Recall: [0.2 0.2 0.4]
F1 Score: [0.18181818 0.2        0.44444444]
```

```
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, Dense,
Dropout, Flatten

count = 0
dirs = os.listdir(r'C:\Users\ROG STRIX\Documents\000000000000000
MESIN\Projek UAS PMDPM_A_Pandas\train_data')
for dir in dirs:
    files = list(os.listdir(r'C:\Users\ROG STRIX\Documents\
000000000000000 MESIN\Projek UAS PMDPM_A_Pandas\train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Jeruk Lemon Folder has 100 Images
Jeruk Nipis Folder has 100 Images
Jeruk Sunkist Folder has 100 Images
Images Folder has 300 Images
```

```
base_dir = r'C:\Users\ROG STRIX\Documents\000000000000000 MESIN\Projek
UAS PMDPM_A_Pandas\train_data'
img_size = 180
batch = 32
validation_split = 0.1

dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
    validation_split=validation_split,
    subset="training",
)
```

```
Found 300 files belonging to 3 classes.

Using 270 files for training.
```

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
```

```
    batch_size=batch,
    validation_split=validation_split,
    subset="validation",
)
```

```
Found 300 files belonging to 3 classes.
Using 30 files for validation.
```

```
class_names = dataset.class_names
print("Class Names:", class_names)
```

```
Class Names: ['Jeruk Lemon', 'Jeruk Nipis', 'Jeruk Sunkist']
```

```
total_count = len(dataset)
val_count = len(val_ds)
train_count = total_count - val_count

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

```
Total Images: 9
Train Images: 8
Validation Images: 1
```

```
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)
```

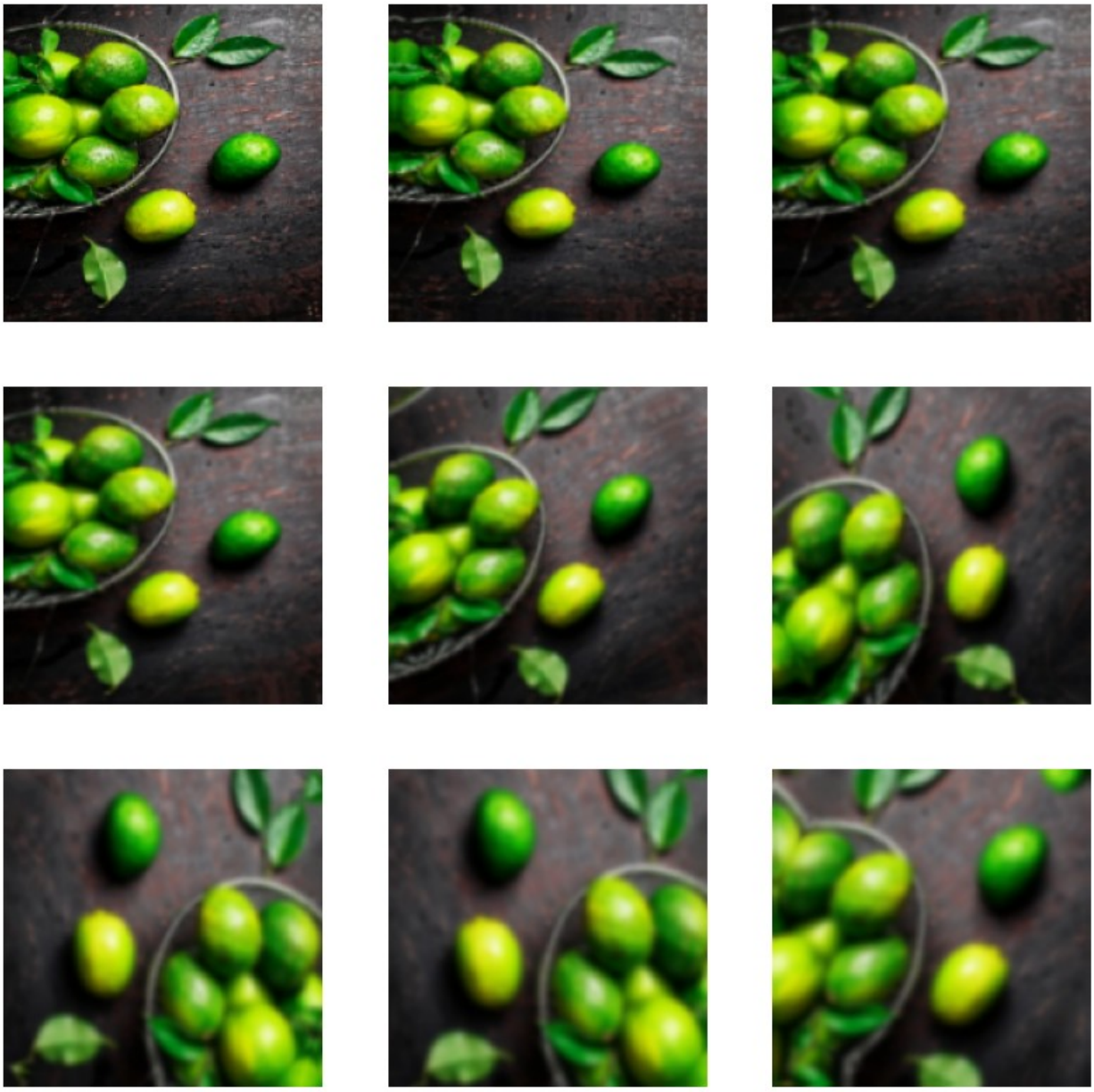
```
(32, 180, 180, 3)
```

```
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)
])
```

```
d:\anaconda3\Lib\site-packages\keras\src\layers\preprocessing\
tf_data_layer.py:19: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
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')
```



```
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model

model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(img_size, img_size, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
```

```
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(len(class_names), activation='softmax')
])
```

```
d:\anaconda3\Lib\site-packages\keras\src\layers\convolutional\
base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
from tensorflow.keras.optimizers import Adam

model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)


model.summary()
```

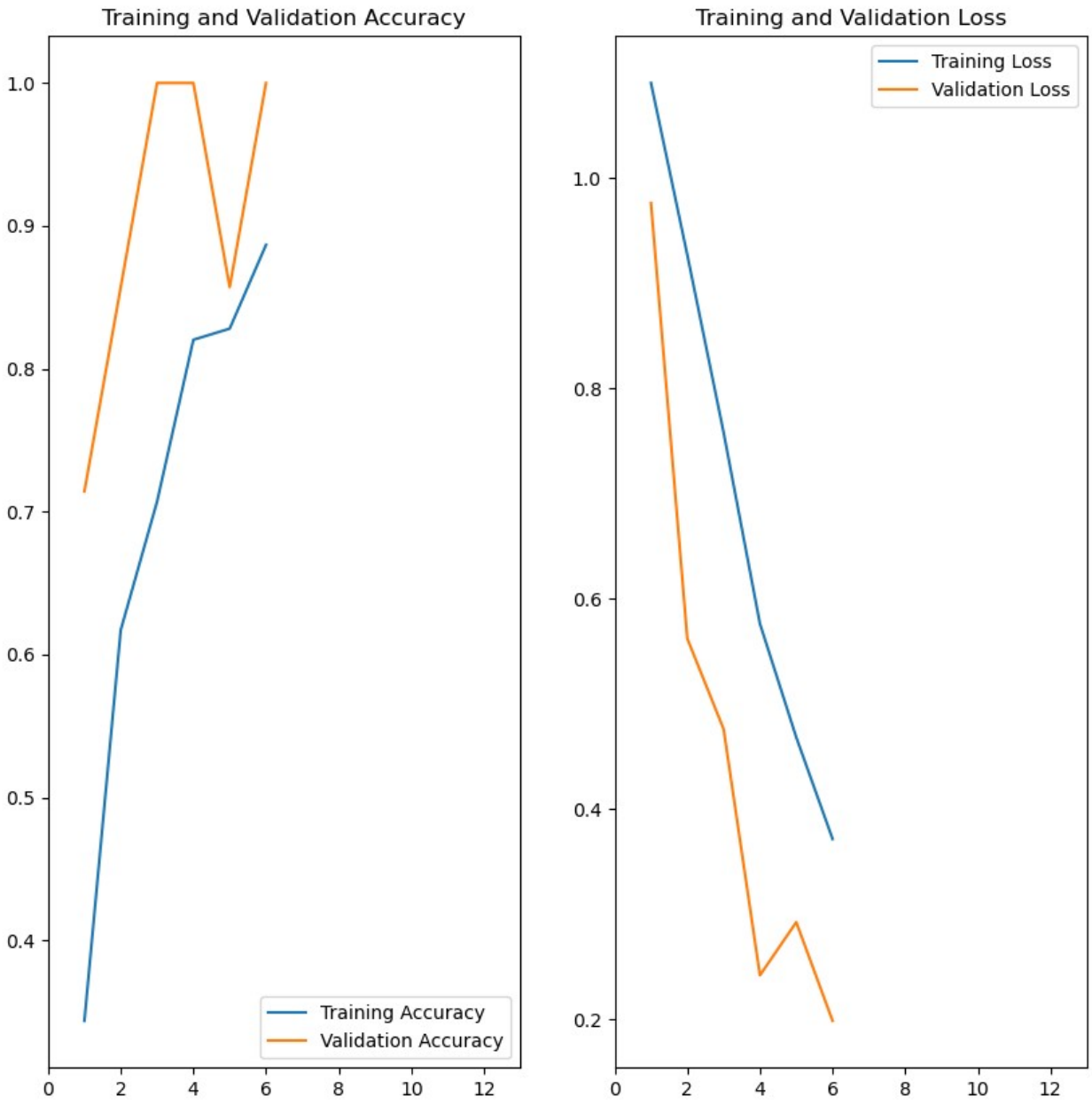
```
Model: "sequential_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━
┩
│ sequential (Sequential)              │ (None, 180, 180, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
│ rescaling (Rescaling)                │ (None, 180, 180, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
│ conv2d (Conv2D)                      │ (None, 178, 178, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
│ max_pooling2d (MaxPooling2D)         │ (None, 89, 89, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
```

```
| conv2d_1 (Conv2D)                    | (None, 87, 87, 64)      |
18,496 |
├──────────────────────────────────────┼─────────────────────────┼──────
──────┤
| max_pooling2d_1 (MaxPooling2D)       | (None, 43, 43, 64)      |
0 |
├──────────────────────────────────────┼─────────────────────────┼──────
──────┤
| conv2d_2 (Conv2D)                    | (None, 41, 41, 128)     |
73,856 |
├──────────────────────────────────────┼─────────────────────────┼──────
──────┤
| max_pooling2d_2 (MaxPooling2D)       | (None, 20, 20, 128)     |
0 |
├──────────────────────────────────────┼─────────────────────────┼──────
──────┤
| flatten (Flatten)                    | (None, 51200)           |
0 |
├──────────────────────────────────────┼─────────────────────────┼──────
──────┤
| dense (Dense)                        | (None, 128)             |
6,553,728 |
├──────────────────────────────────────┼─────────────────────────┼──────
──────┤
| dropout (Dropout)                    | (None, 128)             |
0 |
├──────────────────────────────────────┼─────────────────────────┼──────
──────┤
| dense_1 (Dense)                      | (None, 3)               |
387 |
└──────────────────────────────────────┴─────────────────────────┴──────
──────┘

 Total params: 6,647,363 (25.36 MB)

 Trainable params: 6,647,363 (25.36 MB)

 Non-trainable params: 0 (0.00 B)
```

```
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=3,
                               mode='max')


history= model.fit(train_ds,
                   epochs=30,
                   validation_data=val_ds,
                   callbacks=[early_stopping])
```

```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 404ms/step - accuracy: 0.3287 - loss:
1.0991 - val_accuracy: 0.7143 - val_loss: 0.9764
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 233ms/step - accuracy: 0.6122 - loss:
0.9569 - val_accuracy: 0.8571 - val_loss: 0.5621
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 217ms/step - accuracy: 0.6696 - loss:
0.8141 - val_accuracy: 1.0000 - val_loss: 0.4761
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 229ms/step - accuracy: 0.8408 - loss:
0.5958 - val_accuracy: 1.0000 - val_loss: 0.2419
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 211ms/step - accuracy: 0.8329 - loss:
0.4637 - val_accuracy: 0.8571 - val_loss: 0.2924
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 218ms/step - accuracy: 0.8667 - loss:
0.4076 - val_accuracy: 1.0000 - val_loss: 0.1986


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.xlim(0, 13)
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.xlim(0, 13)
plt.title('Training and Validation Loss')
plt.show()
```




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

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

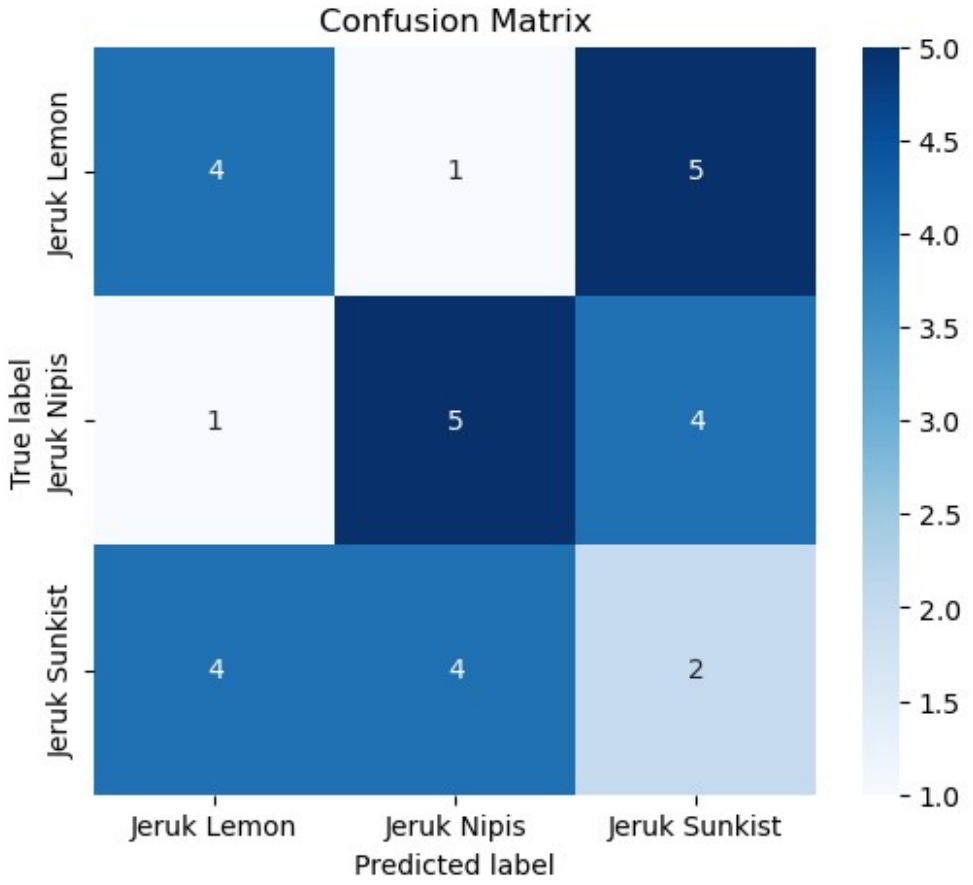
```
from tensorflow.keras.models import load_model
from PIL import Image


model = load_model(r'C:\Users\ROG STRIX\Documents\000000000000000
MESIN\Projek UAS PMDPM_A_Pandas\BestModel_MobileNet_Pandas.h5')
class_names = ['Jeruk Lemon', 'Jeruk Nipis', 'Jeruk Sunkist']

def classify_images(image_path, save_path='predicted_image.jpg'):
    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 = 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'test_data/Jeruk
Sunkist/jeruk_sunkist_test_5.jpg', save_path='sunkist.jpg')
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step
Prediksi: Jeruk Sunkist
Confidence: 54.42%
Prediksi: Jeruk Sunkist dengan confidence 54.42%. Gambar asli disimpan
di sunkist.jpg.
```

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

mobileNet_model = load_model(r'C:\Users\ROG STRIX\Documents\
```

```
000000000000000 MESIN\Projek UAS PMDPM_A_Pandas\
BestModel_MobileNet_Pandas.h5')

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)


y_pred = mobileNet_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)

f1_score = 2 * (precision * recall) / (precision + recall)

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',

            xticklabels=["Jeruk Lemon", "Jeruk Nipis", "Jeruk
Sunkist"], yticklabels=["Jeruk Lemon", "Jeruk Nipis", "Jeruk
Sunkist"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

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())
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 168ms/step
```




```
Confusion Matrix:
 [[4 1 5]
 [1 5 4]
 [4 4 2]]
Akurasi: 0.36666666666666664
Presisi: [0.44444444 0.5        0.18181818]
Recall: [0.4 0.5 0.2]
F1 Score: [0.42105263 0.5        0.19047619]
```

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import load_img,
ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image

base_dir = r'C:\Users\ROG STRIX\Documents\000000000000000 MESIN\Projek
UAS PMDPM_A_Pandas\train_data'
test_data_dir = r'C:\Users\ROG STRIX\Documents\000000000000000 MESIN\
Projek UAS PMDPM_A_Pandas\test_data'

# Parameter pengolahan gambar
img_size = 180
batch_size = 32
validation_split = 0.1
test_split = 0.1

dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch_size,
    validation_split=validation_split + test_split,  # 20% untuk
validasi dan test
    subset="training",
    interpolation="bilinear"
)

# Pembagian lebih lanjut untuk validasi dan test (sisa dari 20%)
val_dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch_size,
    validation_split=validation_split + test_split,
    subset="validation",
    interpolation="bilinear"
)
```

```
Found 300 files belonging to 3 classes.
Using 240 files for training.
Found 300 files belonging to 3 classes.
Using 60 files for validation.
```

```
class_names = dataset.class_names
print("Class Names:", class_names)
```

```
Class Names: ['Jeruk Lemon', 'Jeruk Nipis', 'Jeruk Sunkist']
```

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds =
dataset.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds =
val_dataset.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)
])
```

```
d:\anaconda3\Lib\site-packages\keras\src\layers\preprocessing\
tf_data_layer.py:19: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
def build_vgg16(input_shape=(img_size, img_size, 3)):
    model = models.Sequential([
        # Layer 1: Convolutional Layer 1
        layers.Conv2D(64, (3, 3), activation='relu', padding='same',
input_shape=input_shape),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),

        # Layer 2: Convolutional Layer 2
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),

        # Layer 3: Convolutional Layer 3
        layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),

        # Layer 4: Convolutional Layer 4
        layers.Conv2D(512, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(512, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),

        # Layer 5: Convolutional Layer 5
```

```
        layers.Conv2D(512, (3, 3), activation='relu', padding='same'),
        layers.Conv2D(512, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),

        # Flatten the output of convolutional layers
        layers.Flatten(),

        # Fully Connected Layer 1
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),

        # Fully Connected Layer 2
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),

        # Output Layer
        layers.Dense(len(class_names), activation='softmax')  # 
`len(class_names)` disesuaikan dengan jumlah kelas
    ])

    return model

# Bangun model VGG-16 tanpa pre-trained weights
model_vgg16 = build_vgg16()

# Kompilasi model
model_vgg16.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

```
d:\anaconda3\Lib\site-packages\keras\src\layers\convolutional\
base_conv.py:107: UserWarning: Do not pass an 
`input_shape`/`input_dim` argument to a layer. When using Sequential 
models, prefer using an `Input(shape)` object as the first layer in 
the model instead.
  super().__init__(activity_regularizer=activity_regularizer, 
**kwargs)
```

```
model_vgg16.summary()
```

```
Model: "sequential_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃ 
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 180, 180, 64)        │
```

```
1,792 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_1 (Conv2D)                 | (None, 180, 180, 64)    |
36,928 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| max_pooling2d (MaxPooling2D)      | (None, 90, 90, 64)      |
0 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_2 (Conv2D)                 | (None, 90, 90, 128)     |
73,856 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_3 (Conv2D)                 | (None, 90, 90, 128)     |
147,584 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| max_pooling2d_1 (MaxPooling2D)    | (None, 45, 45, 128)     |
0 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_4 (Conv2D)                 | (None, 45, 45, 256)     |
295,168 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_5 (Conv2D)                 | (None, 45, 45, 256)     |
590,080 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_6 (Conv2D)                 | (None, 45, 45, 256)     |
590,080 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| max_pooling2d_2 (MaxPooling2D)    | (None, 22, 22, 256)     |
0 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_7 (Conv2D)                 | (None, 22, 22, 512)     |
1,180,160 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_8 (Conv2D)                 | (None, 22, 22, 512)     |
2,359,808 |
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| max_pooling2d_3 (MaxPooling2D)    | (None, 11, 11, 512)     |
0 |
```

```
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_9 (Conv2D)                   │ (None, 11, 11, 512)         │
2,359,808 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_10 (Conv2D)                  │ (None, 11, 11, 512)         │
2,359,808 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ max_pooling2d_4 (MaxPooling2D)      │ (None, 5, 5, 512)           │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ flatten (Flatten)                   │ (None, 12800)               │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense (Dense)                       │ (None, 4096)                │
52,432,896 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dropout (Dropout)                   │ (None, 4096)                │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_1 (Dense)                     │ (None, 4096)                │
16,781,312 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dropout_1 (Dropout)                 │ (None, 4096)                │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_2 (Dense)                     │ (None, 3)                   │
12,291 │
└─────────────────────────────────────┴─────────────────────────────┴──────────
──────┘

 Total params: 79,221,571 (302.21 MB)

 Trainable params: 79,221,571 (302.21 MB)

 Non-trainable params: 0 (0.00 B)
```

```
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3,
mode='max')

# Melatih model
history = model_vgg16.fit(
```

```
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=[early_stopping]
)
```

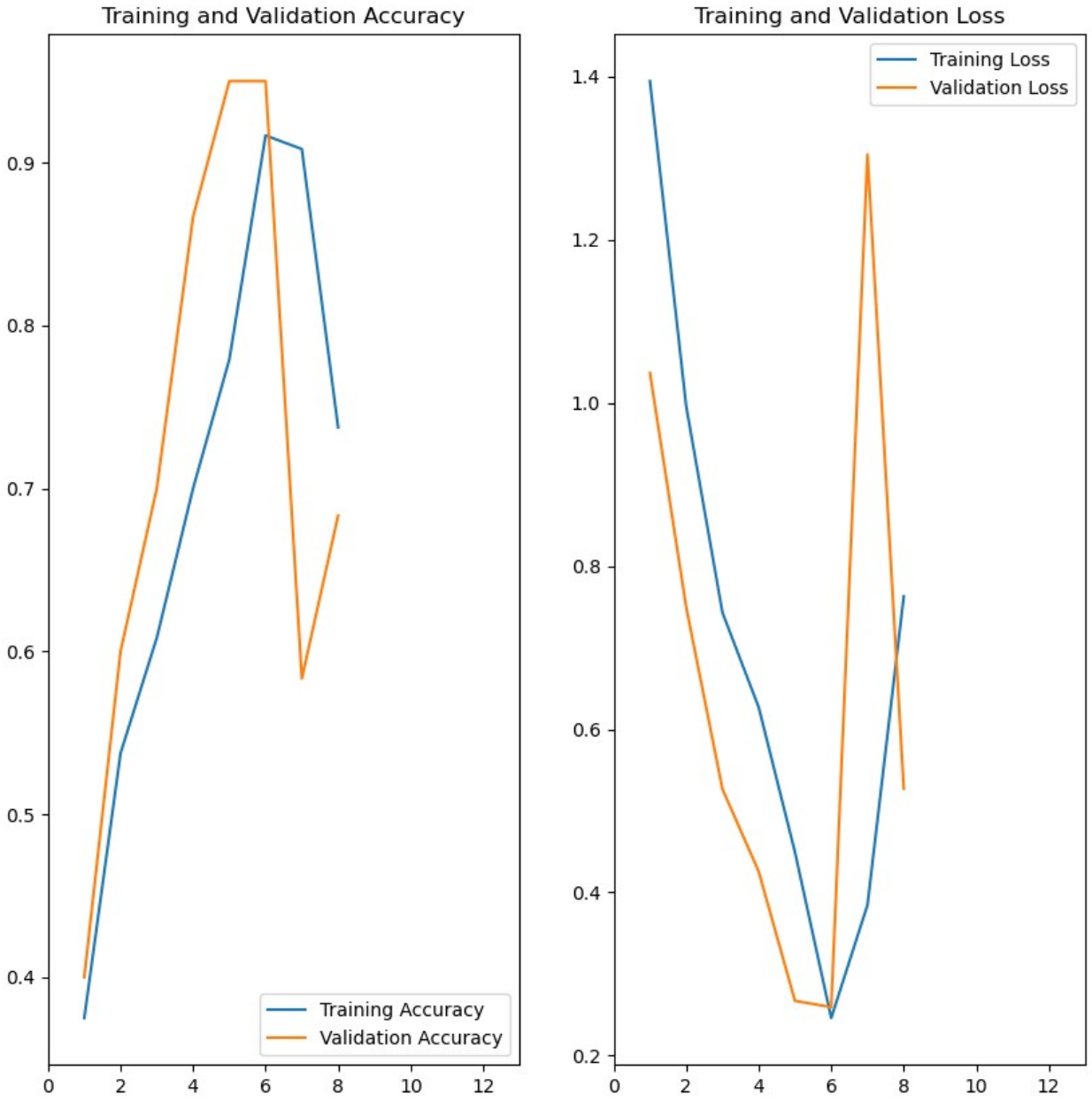
```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 37s 4s/step - accuracy: 0.3841 - loss: 1.5891
- val_accuracy: 0.4000 - val_loss: 1.0366
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 31s 4s/step - accuracy: 0.5402 - loss: 1.0276
- val_accuracy: 0.6000 - val_loss: 0.7496
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 30s 4s/step - accuracy: 0.5614 - loss: 0.8042
- val_accuracy: 0.7000 - val_loss: 0.5266
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 30s 4s/step - accuracy: 0.6885 - loss: 0.6500
- val_accuracy: 0.8667 - val_loss: 0.4257
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 29s 4s/step - accuracy: 0.7575 - loss: 0.4707
- val_accuracy: 0.9500 - val_loss: 0.2672
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 32s 4s/step - accuracy: 0.9299 - loss: 0.2511
- val_accuracy: 0.9500 - val_loss: 0.2594
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 31s 4s/step - accuracy: 0.9421 - loss: 0.2302
- val_accuracy: 0.5833 - val_loss: 1.3042
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 30s 4s/step - accuracy: 0.7535 - loss: 0.9854
- val_accuracy: 0.6833 - val_loss: 0.5272
```

```
model_vgg16.save('BestModel_VGG-16_Pandas.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

```
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.xlim(0, 13)
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.xlim(0, 13)
plt.title('Training and Validation Loss')
plt.show()
```



```
def classify_images(image_path, save_path='predicted_image.jpg'):
    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_vgg16.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'test_data/Jeruk
Lemon/jeruk_lemon_test_8.jpeg', save_path='lemon.jpg')
print(result)
```

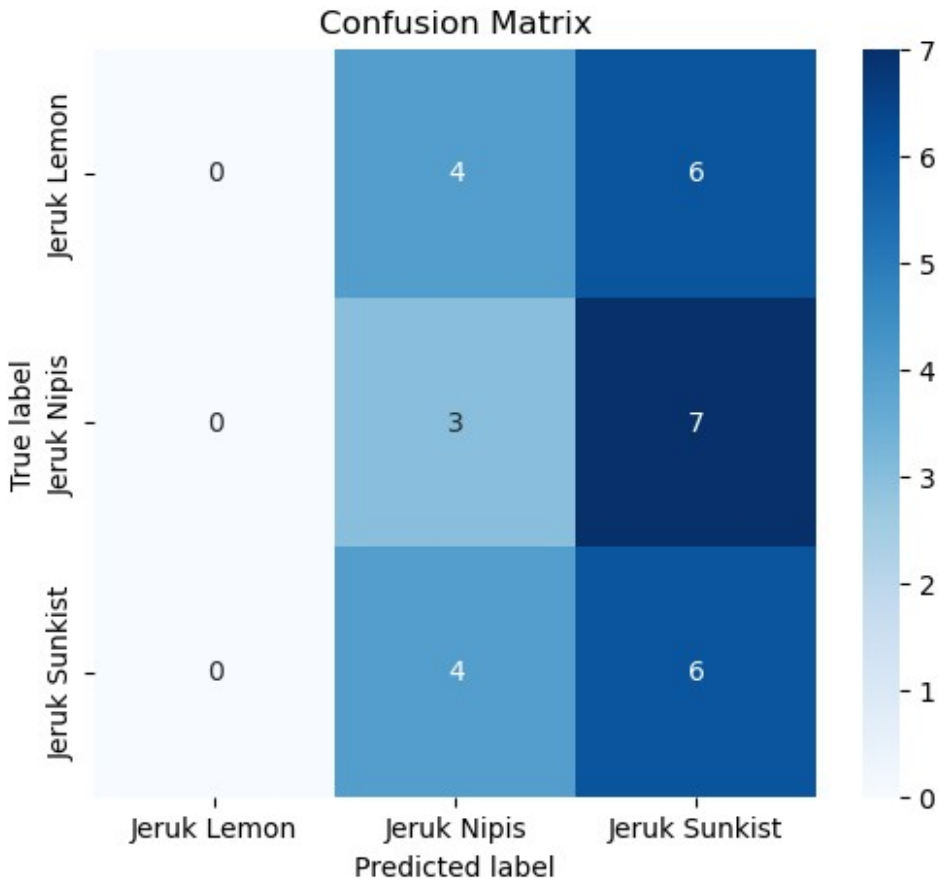
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 200ms/step
Prediksi: Jeruk Sunkist
Confidence: 42.66%
Prediksi: Jeruk Sunkist dengan confidence 42.66%. Gambar asli disimpan
di lemon.jpg.
```

```
from tensorflow.keras.models import load_model

model_vgg16 = load_model('BestModel_VGG-16_Pandas.h5')

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    test_data_dir,
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

y_pred = model_vgg16.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)

f1_score = 2 * (precision * recall) / (precision + recall)

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=test_data.class_names,
yticklabels=test_data.class_names)
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

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())
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

```
Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
```




```
Confusion Matrix:
 [[0 4 6]
 [0 3 7]
 [0 4 6]]
Akurasi: 0.3
Presisi: [       nan 0.27272727 0.31578947]
Recall: [0.  0.3 0.6]
F1 Score: [       nan 0.28571429 0.4137931 ]
```