Gilang Wahyu Nugraha

220711879

Bokeh

Klasifikasi Jenis Buah Anggur (Merah, Thompson, Concord)

MobileNet

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
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\lenovo\OneDrive\Documents\TubesPMDPM\
train_data')
for dir in dirs:
    files = list(os.listdir(r'C:\Users\lenovo\OneDrive\Documents\
TubesPMDPM\train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Concord Folder has 100 Images
Merah Folder has 110 Images
Thompson Folder has 100 Images
Images Folder has 310 Images
```

```
base_dir = r'C:\Users\lenovo\OneDrive\Documents\TubesPMDPM\train_data'
img_size = 180
batch = 16
validation_split = 0.1

num_classes = 3

dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
    class_names=['Merah', 'Thompson', 'Concord']
)
```

```
Found 310 files belonging to 3 classes.
```

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

expected_classes = ['Merah', 'Thompson', 'Concord']
if set(class_names) != set(expected_classes):
    print("Warning: Kelas yang diambil tidak sesuai dengan yang
diharapkan!")
```

```
Class Names: ['Merah', 'Thompson', 'Concord']
```

Pembagian Dataset menjadi Train, Validation, dan Test dengan rasio 80:10:10

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

if train_count < 0 or val_count < 0 or test_count < 0:
    raise ValueError("Jumlah gambar untuk pelatihan, validasi, atau
pengujian tidak valid!")

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

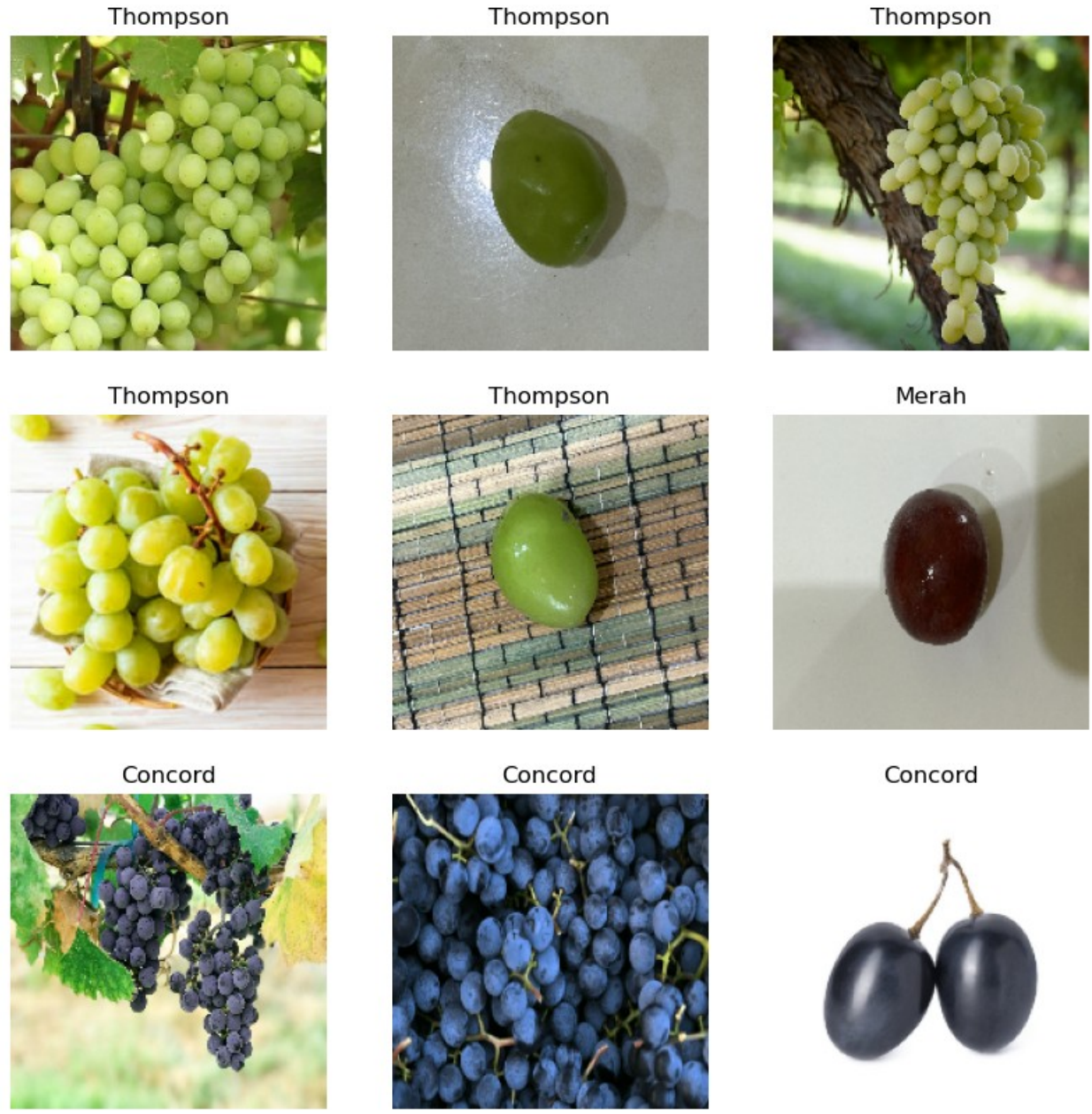
```
Total Images: 20
Train Images: 16
Validation Images: 2
Test Images: 2
```

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

print(f"Dataset telah dibagi menjadi {train_count} gambar untuk
pelatihan dan {val_count} gambar untuk validasi.")
```

```
Dataset telah dibagi menjadi 16 gambar untuk pelatihan dan 2 gambar
untuk validasi.
```

```
import matplotlib.pyplot as plt

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

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

```
        plt.axis('off')

plt.show()
```




```
import numpy as np

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

```
(16, 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),
    layers.RandomBrightness(0.1)
])

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

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

plt.show()
```

```
c:\Users\lenovo\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
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

# Membuat model dari awal
model = Sequential([
    layers.Rescaling(1./255, input_shape=(img_size, img_size, 3)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),
```

```
    layers.BatchNormalization(),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(256, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(len(class_names), activation='softmax')
])

from tensorflow.keras.optimizers import Adam

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

model.summary()
```

Model: "sequential_3"

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ rescaling_1 (Rescaling)              │ (None, 180, 180, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_4 (Conv2D)                    │ (None, 178, 178, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_4                │ (None, 178, 178, 32)        │
128 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_4 (MaxPooling2D)       │ (None, 89, 89, 32)          │
```

```
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_5 (Conv2D)                    | (None, 87, 87, 64)          |
18,496 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| batch_normalization_5                | (None, 87, 87, 64)          |
256 |
| (BatchNormalization)                 |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| max_pooling2d_5 (MaxPooling2D)       | (None, 43, 43, 64)          |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_6 (Conv2D)                    | (None, 41, 41, 128)         |
73,856 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| batch_normalization_6                | (None, 41, 41, 128)         |
512 |
| (BatchNormalization)                 |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| max_pooling2d_6 (MaxPooling2D)       | (None, 20, 20, 128)         |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| conv2d_7 (Conv2D)                    | (None, 18, 18, 256)         |
295,168 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| batch_normalization_7                | (None, 18, 18, 256)         |
1,024 |
| (BatchNormalization)                 |                             |
|
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| max_pooling2d_7 (MaxPooling2D)       | (None, 9, 9, 256)           |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| global_average_pooling2d_1           | (None, 256)                 |
0 |
| (GlobalAveragePooling2D)             |                             |
|
```

```
├─────────────────────────────────┼────────────────────────┼────────
───────┤
│ dense_2 (Dense)                 │ (None, 128)            │
32,896 │
├─────────────────────────────────┼────────────────────────┼────────
───────┤
│ dropout_1 (Dropout)             │ (None, 128)            │
0 │
├─────────────────────────────────┼────────────────────────┼────────
───────┤
│ dense_3 (Dense)                 │ (None, 3)              │
387 │
└─────────────────────────────────┴────────────────────────┴────────
───────┘

 Total params: 423,619 (1.62 MB)

 Trainable params: 422,659 (1.61 MB)

 Non-trainable params: 960 (3.75 KB)
```

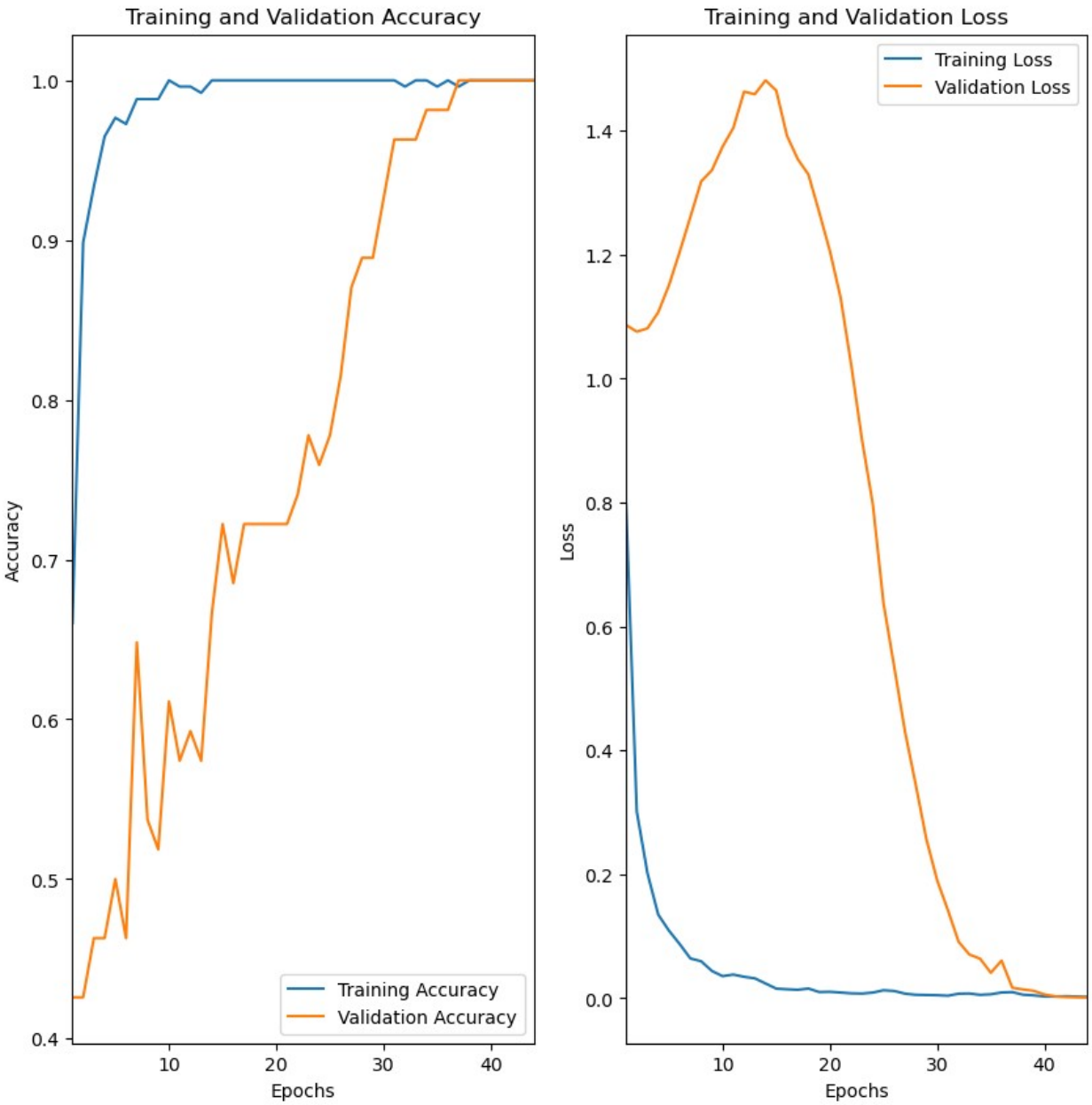
```
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=7,
                               mode='max')

history = model.fit(train_ds,
                    epochs=50,
                    validation_data=val_ds,
                    callbacks=[early_stopping])
```

```
Epoch 1/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 13s 555ms/step - accuracy: 0.5065 - loss:
1.1305 - val_accuracy: 0.4259 - val_loss: 1.0868
Epoch 2/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 443ms/step - accuracy: 0.8666 - loss:
0.3589 - val_accuracy: 0.4259 - val_loss: 1.0759
Epoch 3/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 443ms/step - accuracy: 0.9323 - loss:
0.1850 - val_accuracy: 0.4630 - val_loss: 1.0813
Epoch 4/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 447ms/step - accuracy: 0.9626 - loss:
0.1353 - val_accuracy: 0.4630 - val_loss: 1.1073
Epoch 5/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 443ms/step - accuracy: 0.9665 - loss:
0.1258 - val_accuracy: 0.5000 - val_loss: 1.1509
Epoch 6/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 442ms/step - accuracy: 0.9786 - loss:
0.0842 - val_accuracy: 0.4630 - val_loss: 1.2048
Epoch 7/50
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 445ms/step - accuracy: 0.9957 - loss:
0.0533 - val_accuracy: 0.6481 - val_loss: 1.2614
Epoch 8/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 449ms/step - accuracy: 0.9832 - loss:
0.0667 - val_accuracy: 0.5370 - val_loss: 1.3185
Epoch 9/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 448ms/step - accuracy: 0.9854 - loss:
0.0419 - val_accuracy: 0.5185 - val_loss: 1.3365
Epoch 10/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 449ms/step - accuracy: 1.0000 - loss:
0.0371 - val_accuracy: 0.6111 - val_loss: 1.3747
Epoch 11/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 445ms/step - accuracy: 0.9973 - loss:
0.0311 - val_accuracy: 0.5741 - val_loss: 1.4049
Epoch 12/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 444ms/step - accuracy: 0.9941 - loss:
0.0399 - val_accuracy: 0.5926 - val_loss: 1.4633
Epoch 13/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 445ms/step - accuracy: 0.9918 - loss:
0.0379 - val_accuracy: 0.5741 - val_loss: 1.4593
Epoch 14/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 443ms/step - accuracy: 1.0000 - loss:
0.0221 - val_accuracy: 0.6667 - val_loss: 1.4816
Epoch 15/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 444ms/step - accuracy: 1.0000 - loss:
0.0161 - val_accuracy: 0.7222 - val_loss: 1.4652
Epoch 16/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 442ms/step - accuracy: 1.0000 - loss:
0.0146 - val_accuracy: 0.6852 - val_loss: 1.3921
Epoch 17/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 441ms/step - accuracy: 1.0000 - loss:
0.0135 - val_accuracy: 0.7222 - val_loss: 1.3546
Epoch 18/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 442ms/step - accuracy: 1.0000 - loss:
0.0134 - val_accuracy: 0.7222 - val_loss: 1.3295
Epoch 19/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 444ms/step - accuracy: 1.0000 - loss:
0.0092 - val_accuracy: 0.7222 - val_loss: 1.2690
Epoch 20/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 445ms/step - accuracy: 1.0000 - loss:
0.0096 - val_accuracy: 0.7222 - val_loss: 1.2061
Epoch 21/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 450ms/step - accuracy: 1.0000 - loss:
0.0095 - val_accuracy: 0.7222 - val_loss: 1.1308
Epoch 22/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 446ms/step - accuracy: 1.0000 - loss:
0.0082 - val_accuracy: 0.7407 - val_loss: 1.0207
Epoch 23/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 440ms/step - accuracy: 1.0000 - loss:
```

```
0.0087 - val_accuracy: 0.7778 - val_loss: 0.9006
Epoch 24/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 443ms/step - accuracy: 1.0000 - loss:
0.0079 - val_accuracy: 0.7593 - val_loss: 0.7974
Epoch 25/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 448ms/step - accuracy: 1.0000 - loss:
0.0112 - val_accuracy: 0.7778 - val_loss: 0.6371
Epoch 26/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 447ms/step - accuracy: 1.0000 - loss:
0.0108 - val_accuracy: 0.8148 - val_loss: 0.5343
Epoch 27/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 440ms/step - accuracy: 1.0000 - loss:
0.0080 - val_accuracy: 0.8704 - val_loss: 0.4304
Epoch 28/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 446ms/step - accuracy: 1.0000 - loss:
0.0056 - val_accuracy: 0.8889 - val_loss: 0.3447
Epoch 29/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 440ms/step - accuracy: 1.0000 - loss:
0.0062 - val_accuracy: 0.8889 - val_loss: 0.2561
Epoch 30/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 443ms/step - accuracy: 1.0000 - loss:
0.0051 - val_accuracy: 0.9259 - val_loss: 0.1902
Epoch 31/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 8s 474ms/step - accuracy: 1.0000 - loss:
0.0038 - val_accuracy: 0.9630 - val_loss: 0.1421
Epoch 32/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 8s 479ms/step - accuracy: 0.9984 - loss:
0.0053 - val_accuracy: 0.9630 - val_loss: 0.0910
Epoch 33/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 466ms/step - accuracy: 1.0000 - loss:
0.0083 - val_accuracy: 0.9630 - val_loss: 0.0706
Epoch 34/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 452ms/step - accuracy: 1.0000 - loss:
0.0066 - val_accuracy: 0.9815 - val_loss: 0.0638
Epoch 35/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 437ms/step - accuracy: 0.9973 - loss:
0.0056 - val_accuracy: 0.9815 - val_loss: 0.0410
Epoch 36/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 435ms/step - accuracy: 1.0000 - loss:
0.0105 - val_accuracy: 0.9815 - val_loss: 0.0609
Epoch 37/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 450ms/step - accuracy: 0.9929 - loss:
0.0143 - val_accuracy: 1.0000 - val_loss: 0.0170
Epoch 38/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 445ms/step - accuracy: 1.0000 - loss:
0.0070 - val_accuracy: 1.0000 - val_loss: 0.0141
Epoch 39/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 443ms/step - accuracy: 1.0000 - loss:
0.0065 - val_accuracy: 1.0000 - val_loss: 0.0119
```

```
Epoch 40/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 447ms/step - accuracy: 1.0000 - loss:
0.0023 - val_accuracy: 1.0000 - val_loss: 0.0061
Epoch 41/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 438ms/step - accuracy: 1.0000 - loss:
0.0025 - val_accuracy: 1.0000 - val_loss: 0.0031
Epoch 42/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 437ms/step - accuracy: 1.0000 - loss:
0.0035 - val_accuracy: 1.0000 - val_loss: 0.0019
Epoch 43/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 440ms/step - accuracy: 1.0000 - loss:
0.0023 - val_accuracy: 1.0000 - val_loss: 0.0016
Epoch 44/50
16/16 ━━━━━━━━━━━━━━━━━━━━ 7s 437ms/step - accuracy: 1.0000 - loss:
0.0017 - val_accuracy: 1.0000 - val_loss: 0.0015

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(1, len(epochs_range))
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
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(1, len(epochs_range))
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.show()
```




```
model.save('MobileNet_Bokeh.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\lenovo\OneDrive\Documents\TubesPMDPM\
MobileNet_Bokeh.h5')
class_names = ['Merah', 'Thompson', 'Concord']

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'C:\Users\lenovo\OneDrive\Documents\
TubesPMDPM\test_data\Concord\Concord_Grape_Original_Data010.jpg',
save_path='predicted_image.jpg')
print(result)
```

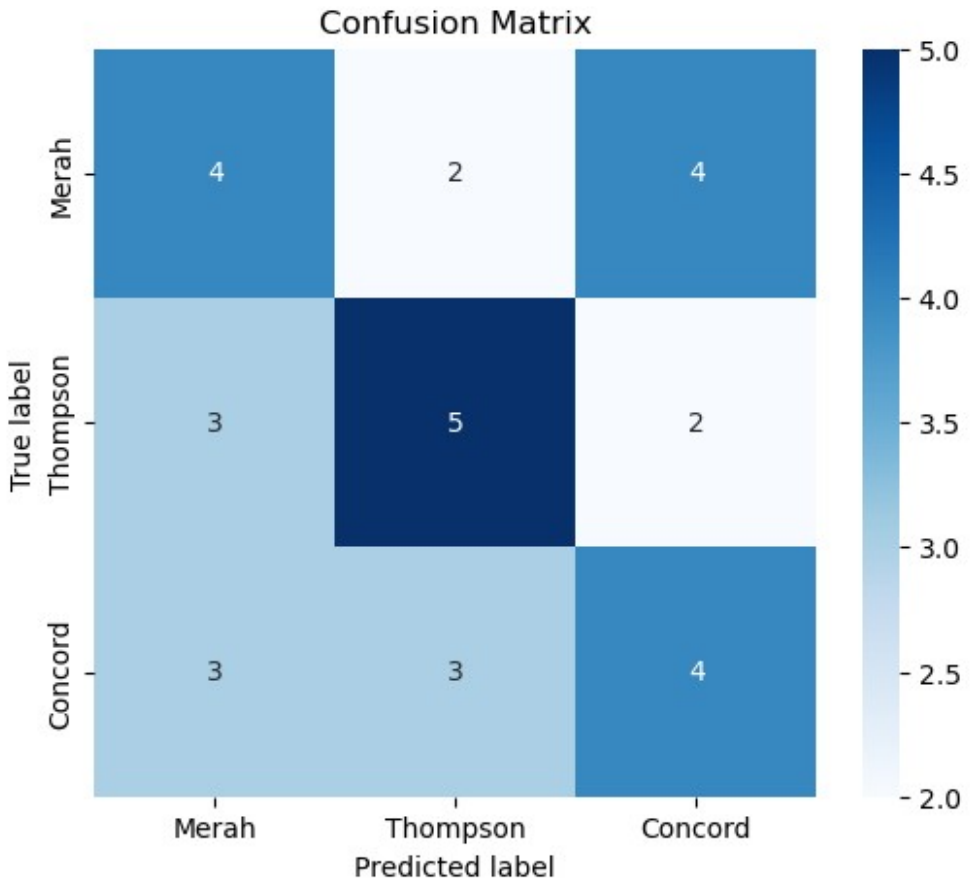
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 162ms/step
Prediksi: Concord
Confidence: 57.43%
Prediksi: Concord dengan confidence 57.43%. Gambar asli disimpan di
predicted_image.jpg.
```

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

model = load_model(r'C:\Users\lenovo\OneDrive\Documents\TubesPMDPM\
MobileNet_Bokeh.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 = 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=["Merah", "Thompson", "Concord"],
yticklabels=["Merah", "Thompson", "Concord"])
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 676ms/step
```




```
Confusion Matrix:
 [[4 2 4]
 [3 5 2]
 [3 3 4]]
Akurasi: 0.43333333333333335
Presisi: [0.4 0.5 0.4]
Recall: [0.4 0.5 0.4]
F1 Score: [0.4 0.5 0.4]
```

Panji Anugrah Agung 220711843 Bokeh Mengklasifikasi Anggur merah, anggur concord, dan anggur Thompson GoogleNet

```
import tensorflow as tf
import cv2
import numpy as np
from matplotlib import pyplot as plt
#load data
data_dir = r"C:\Users\bravo\Downloads\Documents\pembelajaranMesin\
train_data"
#Randomize data yang telah di load sekaligus resize menjadi 180 x 180
data = tf.keras.utils.image_dataset_from_directory(data_dir, seed=
123, image_size=(180,180), batch_size=16)
print(data.class_names)

class_names = data.class_names

img_size = 180
batch = 10
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)
```

```
Found 322 files belonging to 3 classes.
['Concord', 'Merah', 'Thompson']
Found 322 files belonging to 3 classes.
Total Images: 33
Train Images: 30
Validation Images: 3
```

```
import matplotlib.pyplot as plt

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

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


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

```
(10, 180, 180, 3)
```

```
import matplotlib.pyplot as plt

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

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

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

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

```
(10, 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)


#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),
    layers.RandomBrightness(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')
```

```
C:\Users\bravo\AppData\Roaming\Python\Python312\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

#membuat model from scratch
```

```
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
```

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

#Clear Cache Keras menggunakan clear session
K.clear_session()
#buat model dengan
model = googlenet(input_shape, n_classes)
model.summary()
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
```

```
Model: "functional"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)          ┃ Output Shape          ┃       Param # ┃ Connected to          ┃
┡━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━┩
│ input_layer           │ (None, 180, 180,      │             0 │ -                     │
│ (InputLayer)          │ 3)                    │               │                       │
├───────────────────────┼───────────────────────┼───────────────┼───────────────────────┤
│ conv2d (Conv2D)       │ (None, 90, 90,        │         9,472 │ input_layer[0][0]     │
│                       │ 64)                   │               │                       │
├───────────────────────┼───────────────────────┼───────────────┼───────────────────────┤
│ max_pooling2d         │ (None, 45, 45,        │             0 │ conv2d[0][0]          │
│ (MaxPooling2D)        │ 64)                   │               │                       │
├───────────────────────┼───────────────────────┼───────────────┼───────────────────────┤
│ conv2d_1 (Conv2D)     │ (None, 45, 45,        │         4,160 │ max_pooling2d[0][0]…  │
│                       │ 64)                   │               │                       │
├───────────────────────┼───────────────────────┼───────────────┼───────────────────────┤
│ conv2d_2 (Conv2D)     │ (None, 45, 45,        │       110,784 │ conv2d_1[0][0]        │
│                       │ 192)                  │               │                       │
├───────────────────────┼───────────────────────┼───────────────┼───────────────────────┤
│ 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         │ (None, 22, 22,     │           0 │ conv2d_3[0]
[0],   │
```

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

```
──────┤
| 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,    |     19,248 | conv2d_18[0]
[0]   |
```

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

```
──────┤
| 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,    |    295,168 | conv2d_28[0]
[0]   |
```

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

```
──────┤
| 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]… |
```

```
|                       | 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[… |
|                       | 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]… |
| (MaxPooling2D)        |                     |            |
|
|-----------------------|---------------------|-------------|-----------------
------|
| conv2d_46 (Conv2D)    | (None, 6, 6, 160)   |    133,280 |
max_pooling2d_10… |
|-----------------------|---------------------|-------------|-----------------
------|
| conv2d_48 (Conv2D)    | (None, 6, 6, 32)    |     26,656 |
max_pooling2d_10… |
|-----------------------|---------------------|-------------|-----------------
------|
| 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]… │
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
│ conv2d_54 (Conv2D)   │ (None, 6, 6, 48)  │     39,984 │
concatenate_7[0]… │
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
│ max_pooling2d_12     │ (None, 6, 6, 832) │          0 │
concatenate_7[0]… │
│ (MaxPooling2D)       │                      │            │
│
├──────────────────────┼──────────────────────┼─────────────┼──────────────────
──────┤
│ conv2d_51 (Conv2D)   │ (None, 6, 6, 384) │    319,872 │
concatenate_7[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)
```

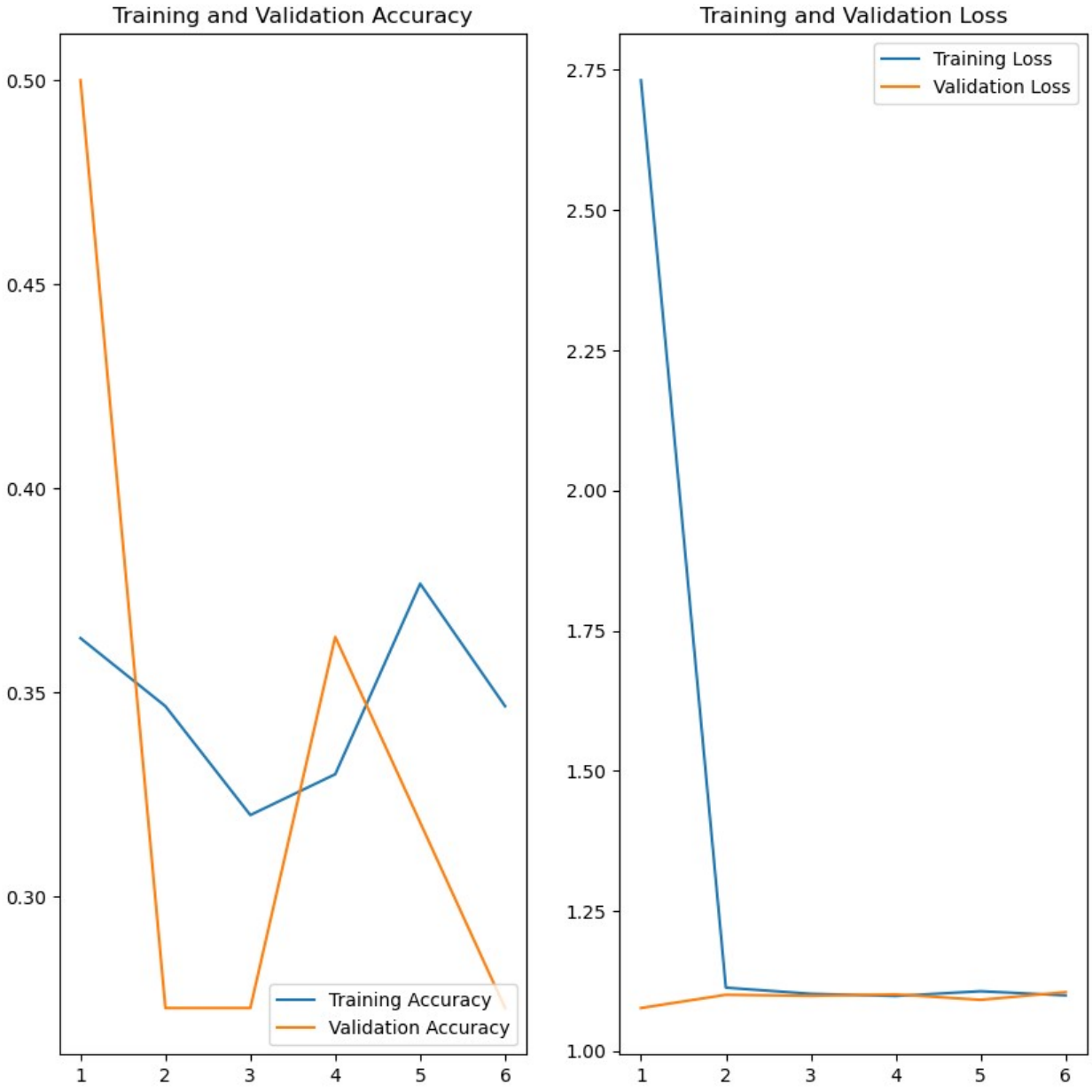
```python
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
#Coimpile dengan optimizer adam
```

```
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
model.compile(
    optimizer=Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

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

```
Epoch 1/30
30/30 ━━━━━━━━━━━━━━━━━━━━ 26s 350ms/step - accuracy: 0.3696 - loss:
5.8330 - val_accuracy: 0.5000 - val_loss: 1.0771
Epoch 2/30
30/30 ━━━━━━━━━━━━━━━━━━━━ 9s 288ms/step - accuracy: 0.3932 - loss:
1.1052 - val_accuracy: 0.2727 - val_loss: 1.1007
Epoch 3/30
30/30 ━━━━━━━━━━━━━━━━━━━━ 9s 286ms/step - accuracy: 0.3212 - loss:
1.1001 - val_accuracy: 0.2727 - val_loss: 1.0987
Epoch 4/30
30/30 ━━━━━━━━━━━━━━━━━━━━ 9s 284ms/step - accuracy: 0.3616 - loss:
1.0981 - val_accuracy: 0.3636 - val_loss: 1.1015
Epoch 5/30
30/30 ━━━━━━━━━━━━━━━━━━━━ 9s 285ms/step - accuracy: 0.3548 - loss:
1.0930 - val_accuracy: 0.3182 - val_loss: 1.0917
Epoch 6/30
30/30 ━━━━━━━━━━━━━━━━━━━━ 9s 286ms/step - accuracy: 0.3872 - loss:
1.0938 - val_accuracy: 0.2727 - val_loss: 1.1054
```

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

plt.subplot(1, 2, 2)
plt.plot(ephocs_range, history.history['loss'], label='Training Loss')
```

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




```
model.save('gugelnet.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
```

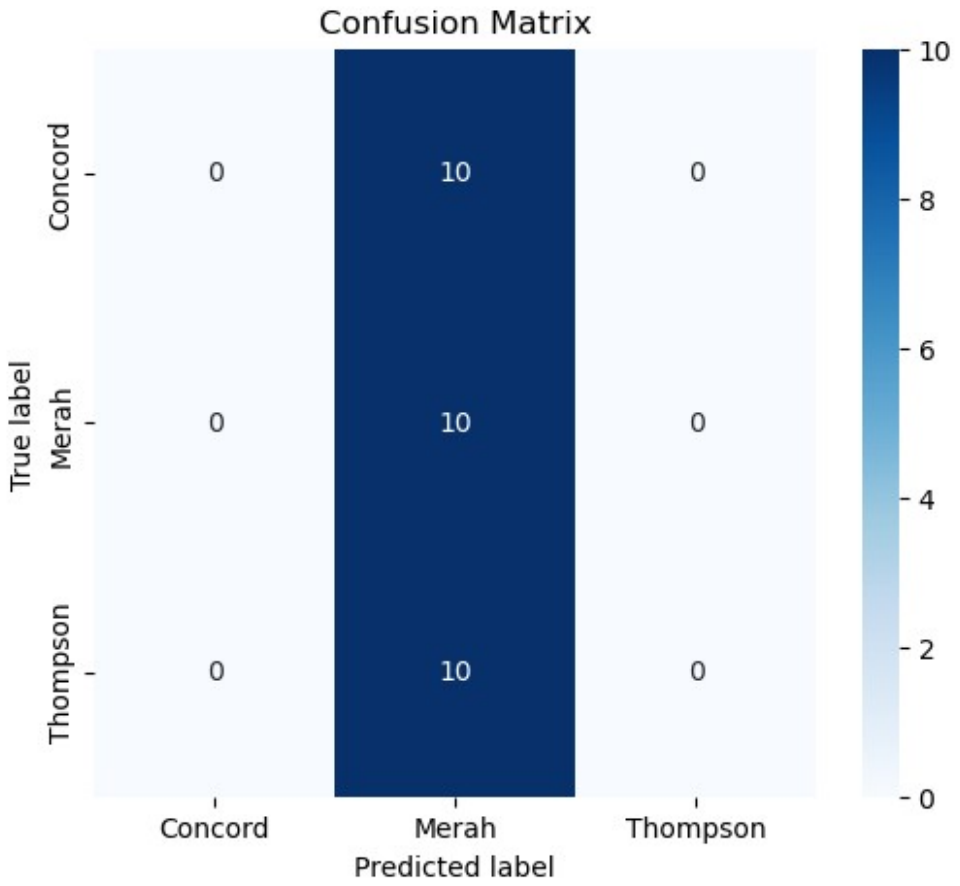
```
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

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

# Load the trained model
model = load_model(r'C:\Users\bravo\Downloads\Documents\
pembelajaranMesin\gugelnet.h5')  # Ganti dengan path model Anda
class_names = ['Concord', 'Merah', 'Thompson']

# 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
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
result = classify_images(r'C:\Users\bravo\Downloads\Documents\
pembelajaranMesin\test_data\Merah\Merah_Grape_Original_Data02.JPG',
save_path='mentah2.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.

WARNING:tensorflow:6 out of the last 6 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000024E79D4D3A0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.

WARNING:tensorflow:6 out of the last 6 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000024E79D4D3A0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 621ms/step
Prediksi: Merah
Confidence: 34.40%
Prediksi: Merah dengan confidence 34.40%. Gambar asli disimpan di mentah2.jpg.
```

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

# Muat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',  # Menghasilkan label dalam bentuk one-hot encoding
    batch_size=32,
    image_size=(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
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
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=["Concord", "Merah", "Thompson"],
yticklabels=["Concord", "Merah", "Thompson"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Menampilkan hasil
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
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 1s/step
```




```
Confusion Matrix:
 [[ 0 10  0]
 [ 0 10  0]
 [ 0 10  0]]
Akurasi: 0.3333333333333333
Presisi: [       nan 0.33333333        nan]
Recall: [0. 1. 0.]
F1 Score: [nan 0.5 nan]
```

Alouisius Gonzaga Duan Allegra / 220711881 / Bokeh / Mengklasifikasikan Anggur (Merah, Thompson, Concord) / Arsitektur VGG-16

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

data_dir = r"C:\Users\smerf\Downloads\Tugas6_B_11881\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

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)
```

```
Found 300 files belonging to 3 classes.
['Concord', 'Merah', 'Thompson']
Found 300 files belonging to 3 classes.
Total Images: 10
Train Images: 9
Validation 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.axis('off')
```

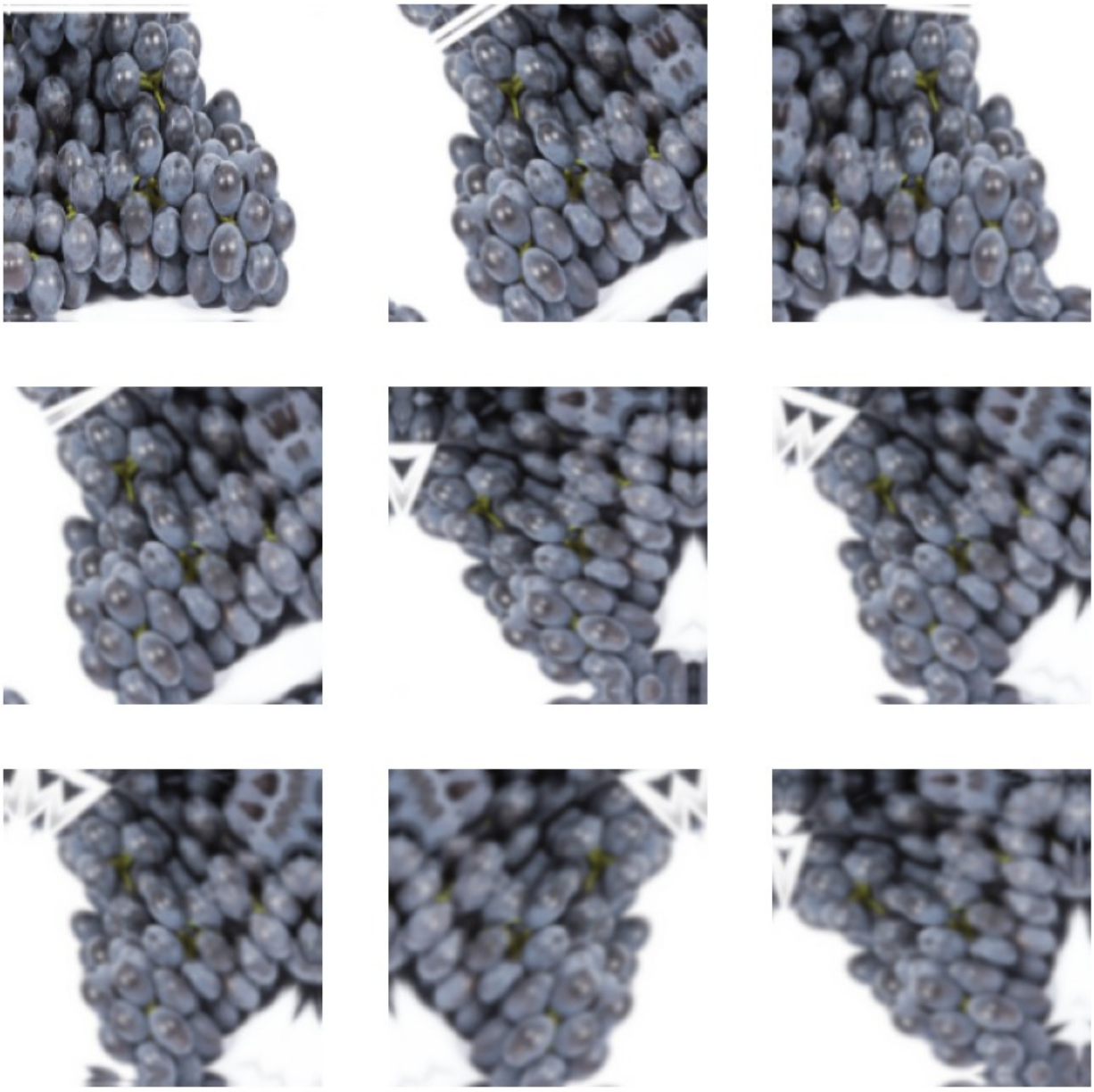
```
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

```
(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)

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')
```

```
C:\Users\smerf\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\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
from tensorflow.keras import layers, models
import keras._tf_keras.keras.backend as K

def vgg16(input_shape, n_classes):
    model = models.Sequential()

    model.add(layers.Conv2D(64, (3, 3), activation='relu',
padding='same', input_shape=input_shape))
    model.add(layers.Conv2D(64, (3, 3), activation='relu',
padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
```

```
    model.add(layers.Conv2D(128, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(128, (3, 3), activation='relu',
padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

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

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

    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(n_classes, activation='softmax'))

    return model

input_shape = (180, 180, 3)
n_classes = 2

K.clear_session()

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

```
WARNING:tensorflow:From C:\Users\smerf\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
```

```
packages\Python311\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.


C:\Users\smerf\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\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: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
┓
┃ 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 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
```

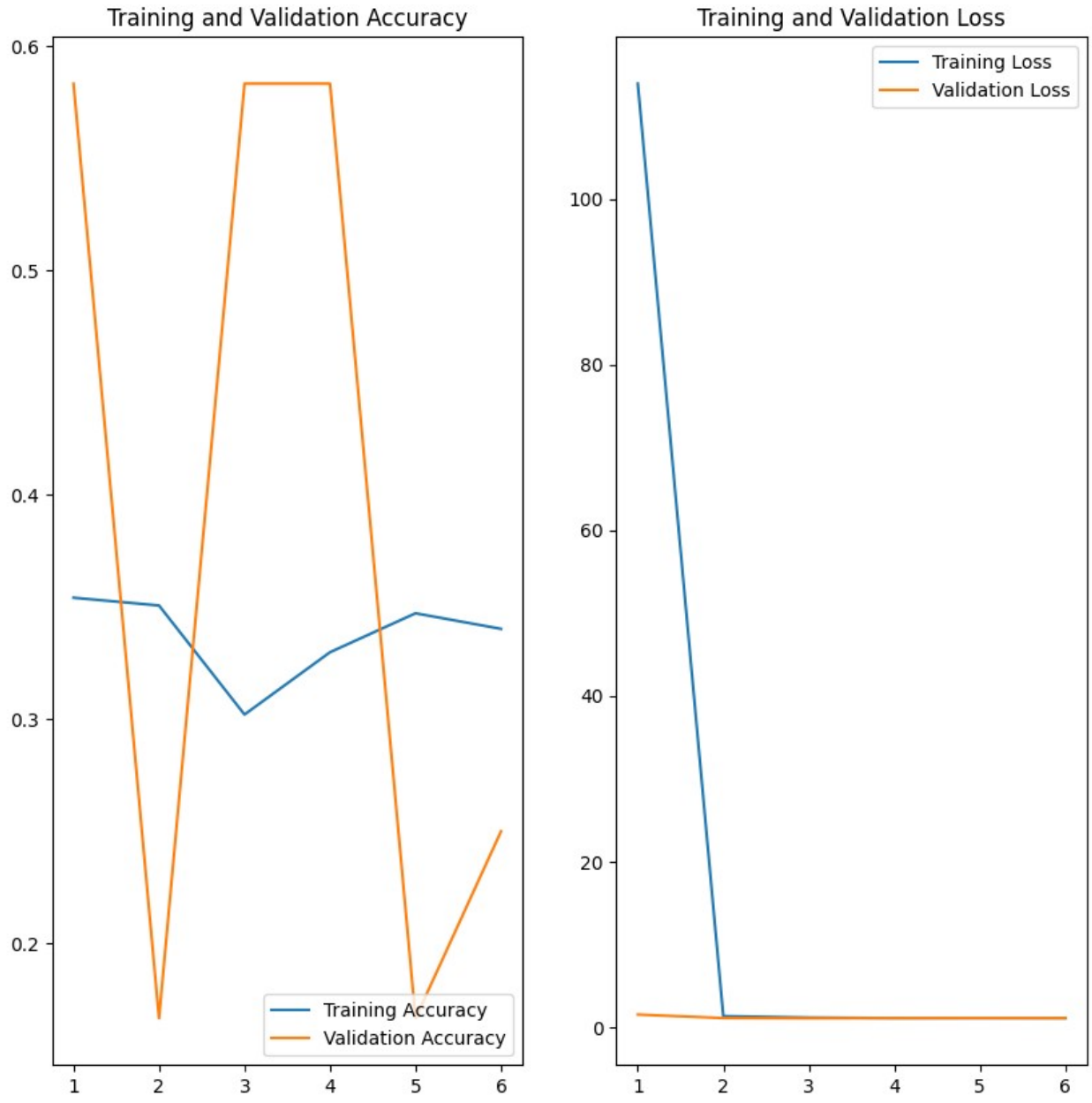
```
──────┤
| conv2d_5 (Conv2D)                | (None, 45, 45, 256)    |
590,080 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_6 (Conv2D)                | (None, 45, 45, 256)    |
590,080 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| 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 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_9 (Conv2D)                | (None, 22, 22, 512)    |
2,359,808 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| max_pooling2d_3 (MaxPooling2D)   | (None, 11, 11, 512)    |
0 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_10 (Conv2D)               | (None, 11, 11, 512)    |
2,359,808 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_11 (Conv2D)               | (None, 11, 11, 512)    |
2,359,808 |
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
| conv2d_12 (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, 2)                 |
8,194 |
└─────────────────────────────────────┴───────────────────────────┴──────────
──────┘

 Total params: 83,937,090 (320.19 MB)

 Trainable params: 83,937,090 (320.19 MB)

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

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

n_classes = 3

model = vgg16(input_shape, n_classes)

model.compile(
    optimizer=Adam(),
    loss='sparse_categorical_crossentropy',
)

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
9/9 ━━━━━━━━━━━━━━━━━━━━ 115s 12s/step - accuracy: 0.3861 - loss:
197.9797 - val_accuracy: 0.5833 - val_loss: 1.5339
```

```
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 105s 12s/step - accuracy: 0.3919 - loss:
1.3644 - val_accuracy: 0.1667 - val_loss: 1.1038
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 109s 12s/step - accuracy: 0.2777 - loss:
1.1617 - val_accuracy: 0.5833 - val_loss: 1.0776
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 106s 12s/step - accuracy: 0.3198 - loss:
1.1015 - val_accuracy: 0.5833 - val_loss: 1.0951
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 100s 11s/step - accuracy: 0.3596 - loss:
1.0998 - val_accuracy: 0.1667 - val_loss: 1.1126
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 107s 12s/step - accuracy: 0.3625 - loss:
1.0999 - val_accuracy: 0.2500 - val_loss: 1.0935

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('vgg-16.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\smerf\Downloads\Tugas6_B_11881\vgg-
16.h5')
class_names = ['Merah', 'Thompson', 'Concord']

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'C:\Users\smerf\Downloads\Tugas6_B_11881\
train_data\Concord\Concord_Grape_Original_Data009.jpg',
save_path='concord9.jpg')
print(result)
```

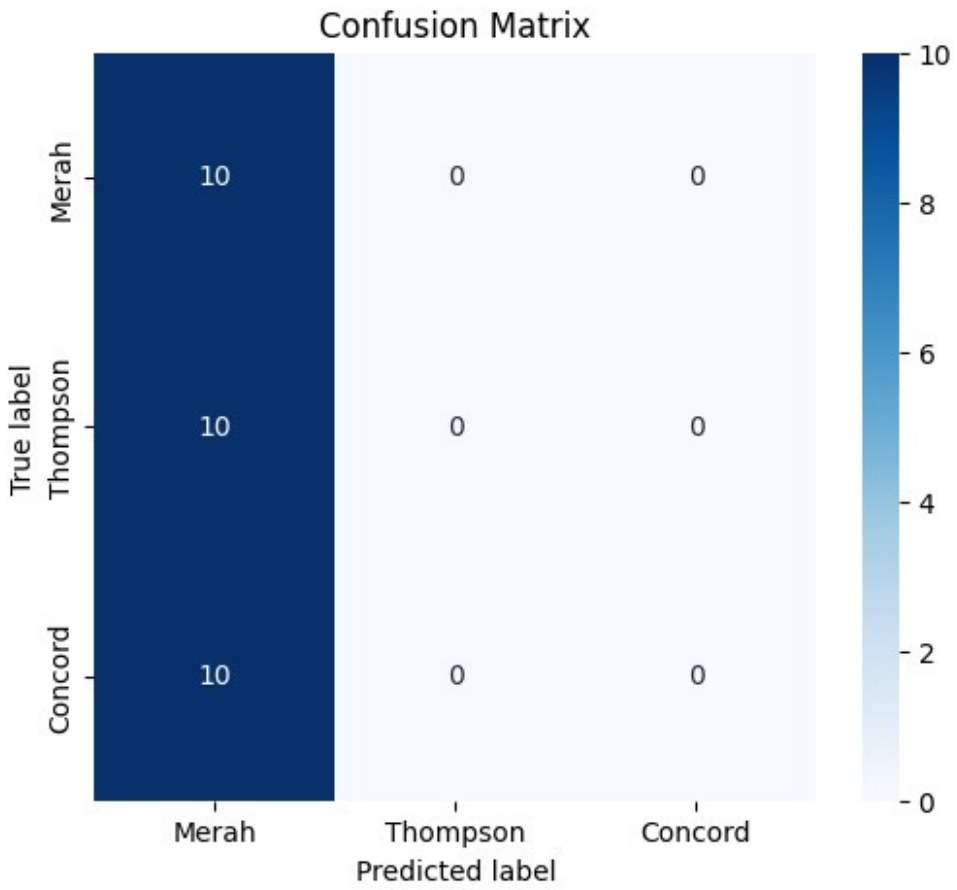
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 406ms/step
Prediksi: Merah
Confidence: 33.49%
Prediksi: Merah dengan confidence 33.49%. Gambar asli disimpan di
concord9.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=["Merah", "Thompson", "Concord"],
            yticklabels=["Merah", "Thompson", "Concord"])
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 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
```




```
Confusion Matrix:
 [[10  0  0]
 [10  0  0]
 [10  0  0]]
Akurasi: 0.3333333333333333
Presisi: [0.33333333        nan        nan]
Recall: [1. 0. 0.]
F1 Score: [0.5 nan nan]
```

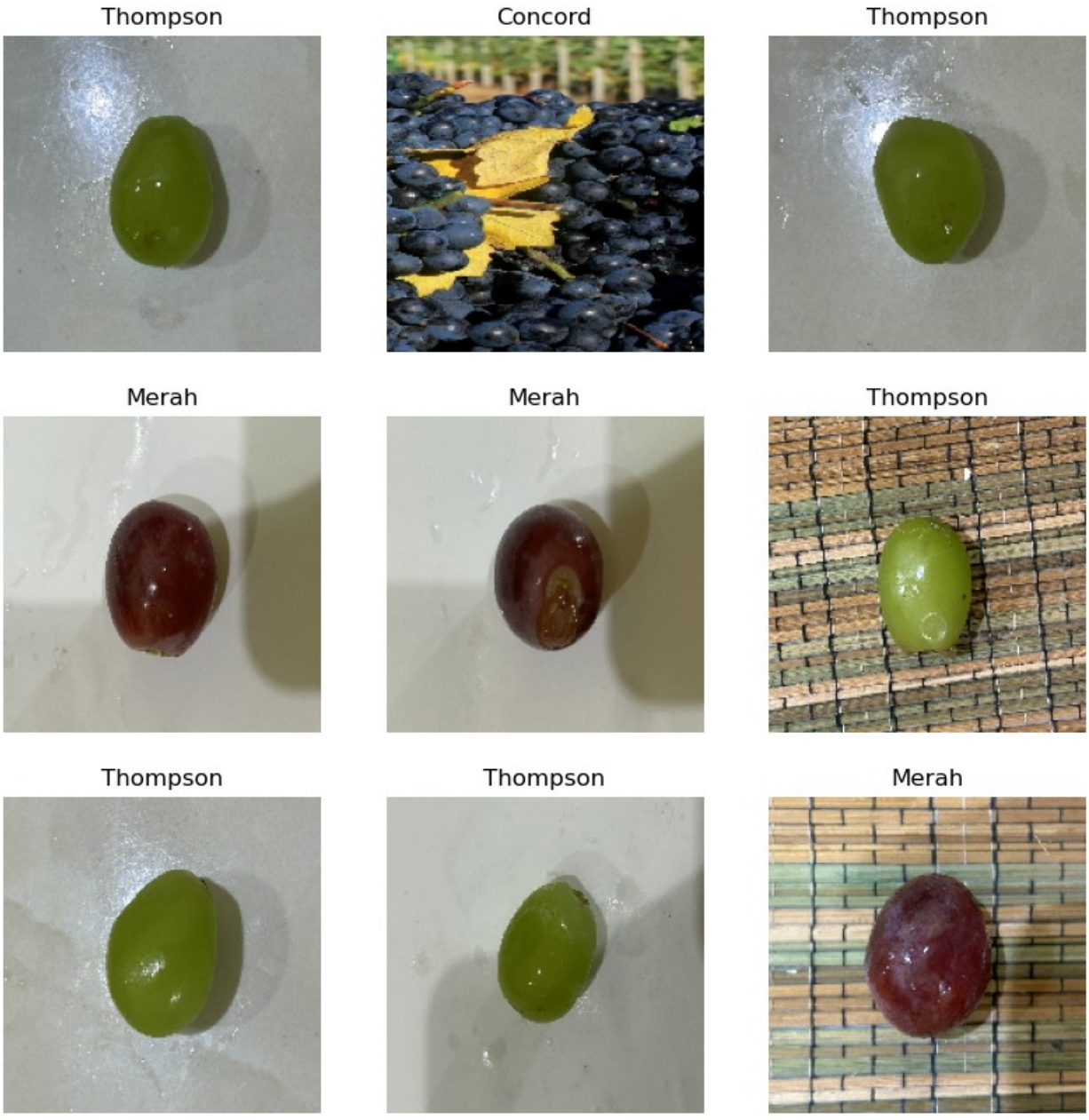
Julius Aditya 220711892 Bokeh Mengklsifikasi Anggur Merah, Concord, dan Thompson

```
import tensorflow as tf
import cv2
import numpy as np
from matplotlib import pyplot as plt
data_dir = r"C:\Users\LENOVO LEGION\Videos\TUBESMLUAS\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

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)
```

```
Found 300 files belonging to 3 classes.
['Concord', 'Merah', 'Thompson']
Found 300 files belonging to 3 classes.
Total Images: 10
Train Images: 9
Validation 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')
```

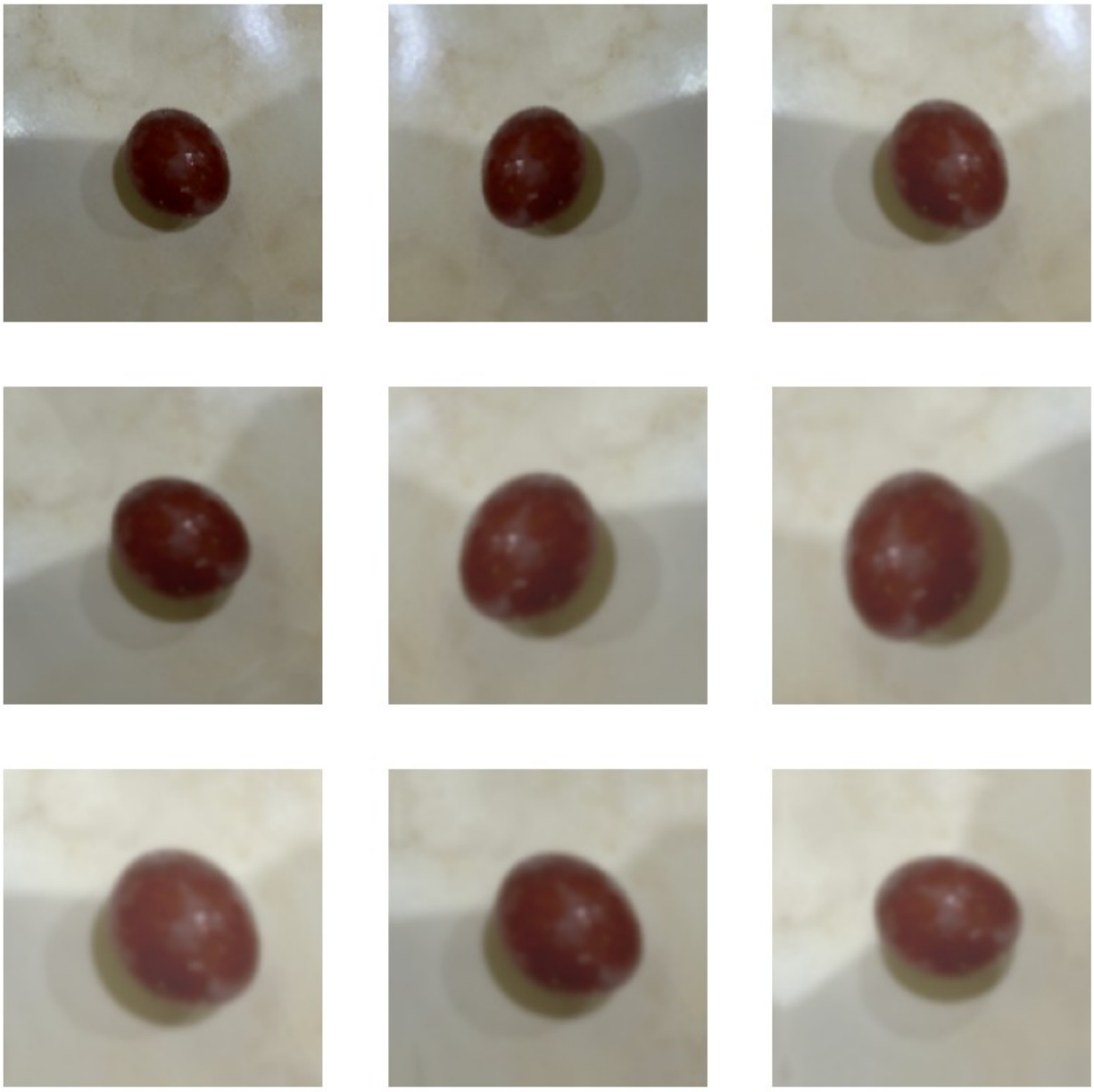



```
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

```
(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)

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



```
import tensorflow as tf
from keras.models import Model
from keras.layers import Input, Conv2D, MaxPool2D, Flatten, Dense,
Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

    x = Conv2D(96, (11, 11), strides=4, activation='relu',
```

```
padding='valid')(input)
    x = MaxPool2D((3, 3), strides=2)(x)

    x = Conv2D(256, (5, 5), activation='relu', padding='same')(x)
    x = MaxPool2D((3, 3), strides=2)(x)

    x = Conv2D(384, (3, 3), activation='relu', padding='same')(x)

    x = Conv2D(384, (3, 3), activation='relu', padding='same')(x)

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

    x = Flatten()(x)

    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(1000, activation='relu')(x)

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

    model = Model(input, output)
    return model

input_shape = (227, 227, 3)
n_classes = 3

model = alexnet(input_shape, n_classes)
model.summary()

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

datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_ds = datagen.flow_from_directory(
    r'C:\Users\LENOVO LEGION\Videos\TUBESMLUAS\train_data',
    target_size=(227, 227),
    batch_size=32,
    class_mode='sparse'
)

val_ds = val_datagen.flow_from_directory(
    r'C:\Users\LENOVO LEGION\Videos\TUBESMLUAS\train_data',
```

```
    target_size=(227, 227),
    batch_size=32,
    class_mode='sparse'
)

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

history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=[early_stopping],
    verbose=1
)
```

```
Model: "functional_11"
```

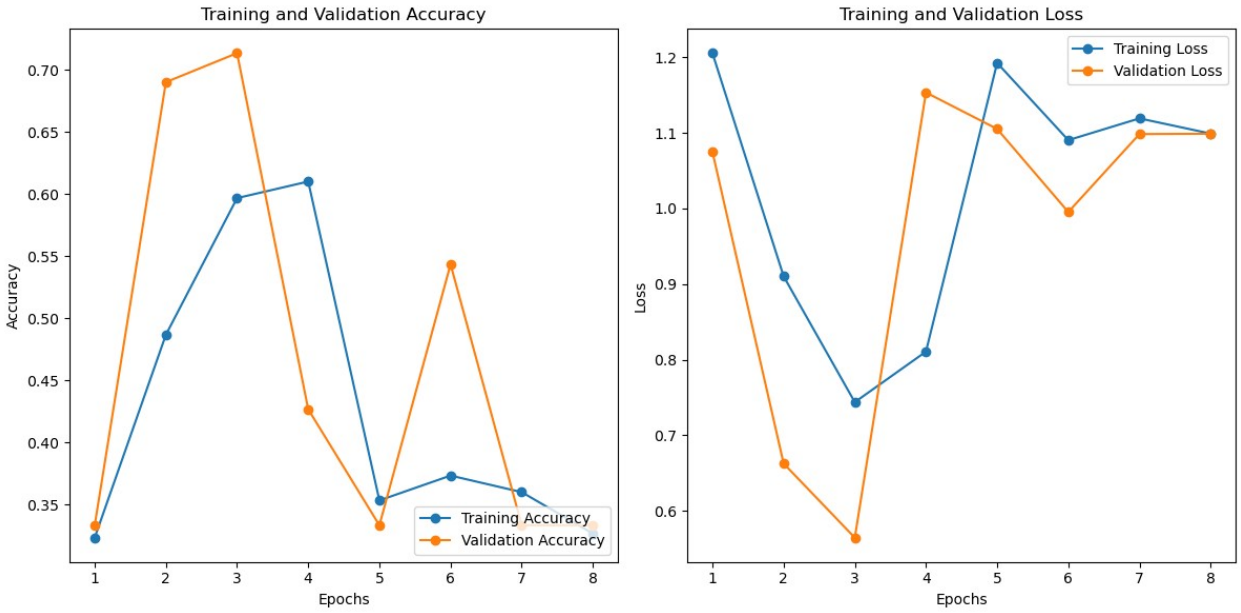
```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ input_layer_7 (InputLayer)           │ (None, 227, 227, 3)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_15 (Conv2D)                   │ (None, 55, 55, 96)          │          34,944 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_9 (MaxPooling2D)       │ (None, 27, 27, 96)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_16 (Conv2D)                   │ (None, 27, 27, 256)         │         614,656 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_10 (MaxPooling2D)      │ (None, 13, 13, 256)         │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_17 (Conv2D)                   │ (None, 13, 13, 384)         │         885,120 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
```

```
──────┤
| conv2d_18 (Conv2D)                | (None, 13, 13, 384)       |
1,327,488 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| conv2d_19 (Conv2D)                | (None, 13, 13, 256)       |
884,992 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| max_pooling2d_11 (MaxPooling2D) | (None, 6, 6, 256)         |
0 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| flatten_3 (Flatten)               | (None, 9216)              |
0 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| dense_12 (Dense)                  | (None, 4096)              |
37,752,832 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| dropout_6 (Dropout)               | (None, 4096)              |
0 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| dense_13 (Dense)                  | (None, 4096)              |
16,781,312 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| dropout_7 (Dropout)               | (None, 4096)              |
0 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| dense_14 (Dense)                  | (None, 1000)              |
4,097,000 |
├──────────────────────────────────┼──────────────────────────────┼──────
──────┤
| dense_15 (Dense)                  | (None, 3)                 |
3,003 |
└──────────────────────────────────┴──────────────────────────────┴──────
──────┘

 Total params: 62,381,347 (237.97 MB)

 Trainable params: 62,381,347 (237.97 MB)

 Non-trainable params: 0 (0.00 B)

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

```
Epoch 1/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 82s 6s/step - accuracy: 0.2957 - loss:
1.2819 - val_accuracy: 0.3333 - val_loss: 1.0748
Epoch 2/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 74s 5s/step - accuracy: 0.4131 - loss:
0.9865 - val_accuracy: 0.6900 - val_loss: 0.6626
Epoch 3/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 74s 5s/step - accuracy: 0.6216 - loss:
0.6762 - val_accuracy: 0.7133 - val_loss: 0.5645
Epoch 4/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 76s 5s/step - accuracy: 0.6399 - loss:
0.7429 - val_accuracy: 0.4267 - val_loss: 1.1534
Epoch 5/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 72s 5s/step - accuracy: 0.3752 - loss:
1.2006 - val_accuracy: 0.3333 - val_loss: 1.1052
Epoch 6/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 77s 6s/step - accuracy: 0.3161 - loss:
1.1026 - val_accuracy: 0.5433 - val_loss: 0.9953
Epoch 7/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 79s 6s/step - accuracy: 0.3993 - loss:
1.1281 - val_accuracy: 0.3333 - val_loss: 1.0984
Epoch 8/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 73s 5s/step - accuracy: 0.2945 - loss:
1.1013 - val_accuracy: 0.3333 - val_loss: 1.0988
```

```python
import matplotlib.pyplot as plt

epochs_range = range(1, len(history.history['loss']) + 1)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, history.history['accuracy'], label='Training
Accuracy', marker='o')
plt.plot(epochs_range, history.history['val_accuracy'],
label='Validation Accuracy', marker='o')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, history.history['loss'], label='Training Loss',
marker='o')
plt.plot(epochs_range, history.history['val_loss'], label='Validation
Loss', marker='o')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.tight_layout()
plt.show()
```




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

```
c:\Users\LENOVO LEGION\anaconda3\Lib\site-packages\keras\src\models\
model.py:342: UserWarning: You are saving your model as an HDF5 file
via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  warnings.warn(
```

```
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\LENOVO LEGION\Videos\TUBESMLUAS\
alexnet.h5')
class_names = ['Concord', 'Merah', 'Thompson']

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\Merah\
Merah_Grape_Original_Data03.JPG', save_path='merah2.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.

Terjadi kesalahan: Input 0 of layer "functional_11" is incompatible
with the layer: expected shape=(None, 227, 227, 3), found shape=(1,
180, 180, 3)
```

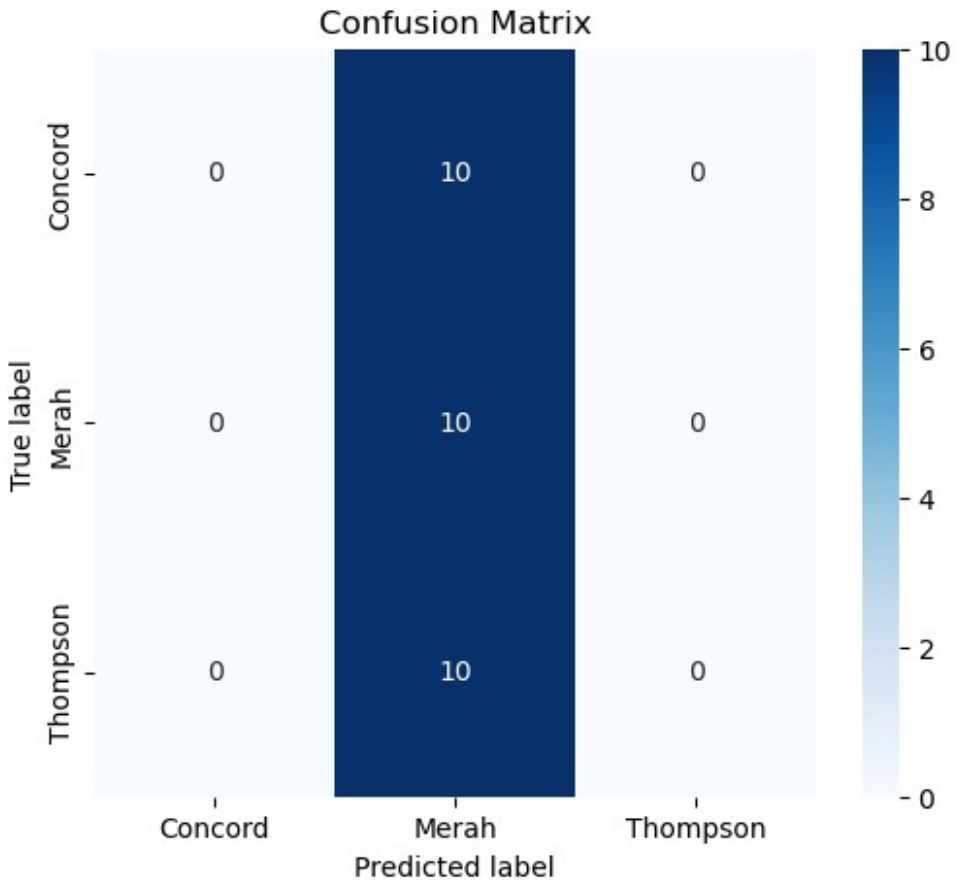
```
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(227, 227)
)

model = load_model(r'C:\Users\LENOVO LEGION\Videos\TUBESMLUAS\
alexnet.h5')

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=["Concord", "Merah", "Thompson"],
yticklabels=["Concord", "Merah", "Thompson"])
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.
```

```
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 827ms/step
```




```
Confusion Matrix:
 [[ 0 10  0]
 [ 0 10  0]
 [ 0 10  0]]
Akurasi:  0.3333333333333333
Presisi:  [       nan 0.33333333        nan]
Recall: [0. 1. 0.]
F1 Score:  [nan 0.5 nan]
```