Mario Arya Mahardika/ 220711664/ Alexnet

In [117…
```
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropou
from tensorflow.keras.preprocessing import image_dataset_from_directory
from matplotlib import pyplot as plt

# Load data
data_dir = r"D:\ATMA\sem 5\Mesin\UAS\train_data"

batch_size = 16
img_height, img_width = 180, 180

data = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)

print(data.class_names)

class_names = data.class_names
```

```
Found 300 files belonging to 3 classes.
['BlackBerry', 'Blueberry', 'strawberry']
```

In [118…
```
img_size = 180
batch = 32
validation_split = 0.1
test_split = 0.1
dataset = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,

)
```

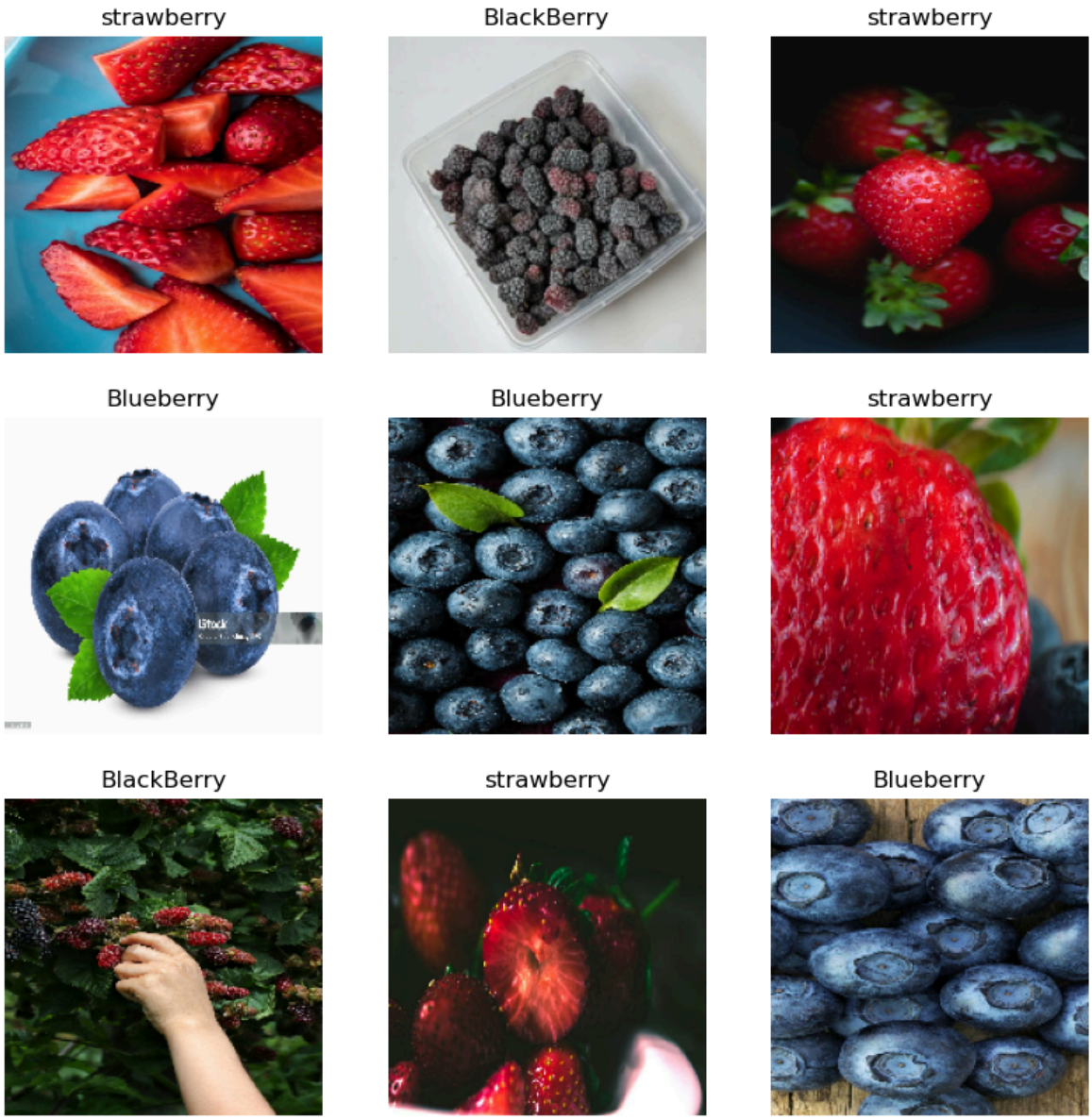
```
Found 300 files belonging to 3 classes.
```

In [119…
```
total_count = len(dataset)
val_count = int(total_count * validation_split)
test_count = int(total_count * test_split)
train_count = total_count - val_count - test_count

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

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

In [120…

```
import matplotlib.pyplot as plt

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


for images, labels in data.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')
```



In [121…

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

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

In [122…

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

Tuner = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).map(
    lambda x, y: (data_augmentation(x), y)
).prefetch(buffer_size=Tuner)

val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=Tuner)


data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 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')
```

```
c:\Users\user\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)
```

In [123…

```
import tensorflow as tf
from tensorflow.keras import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPool2D, Flatten, Dense, Dr


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

    x = Conv2D(96, kernel_size=(11, 11), strides=(4, 4), activation='relu')(inpu
    x = MaxPool2D(pool_size=(3, 3), strides=(2, 2))(x)

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

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

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

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

    x = Flatten()(x)

    x = Dense(4096, activation='relu')(x)
```

```
    x = Dropout(0.5)(x)

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

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

    model = Model(input, output)
    return model

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

tf.keras.backend.clear_session()
model = alexnet(input_shape, n_classes)
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 180, 180, 3)	0
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, 2)	8,194

Total params: 37,318,018 (142.36 MB)

Trainable params: 37,318,018 (142.36 MB)

Non-trainable params: 0 (0.00 B)

In [124…

```
import tensorflow as tf
from tensorflow.keras.models import Model
```

```
from tensorflow.keras.layers import Input, Conv2D, MaxPool2D, Flatten, Dense, Dr
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

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

    x = Conv2D(96, kernel_size=(11, 11), strides=(4, 4), activation='relu')(inpu
    x = MaxPool2D(pool_size=(3, 3), strides=(2, 2))(x)

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

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

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

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

    x = Flatten()(x)

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

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

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

    model = Model(input, output)
    return model

n_classes = len(dataset.class_names)
print("Class names:", dataset.class_names)
print("Number of classes:", n_classes)

train_ds = train_ds.map(lambda x, y: (x, tf.cast(y, tf.int32)))
val_ds = val_ds.map(lambda x, y: (x, tf.cast(y, tf.int32)))

for images, labels in train_ds.take(1):
    print("Unique labels in train dataset:", tf.unique(labels)[0].numpy())

input_shape = (180, 180, 3)
tf.keras.backend.clear_session()
model = alexnet(input_shape, n_classes)

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

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

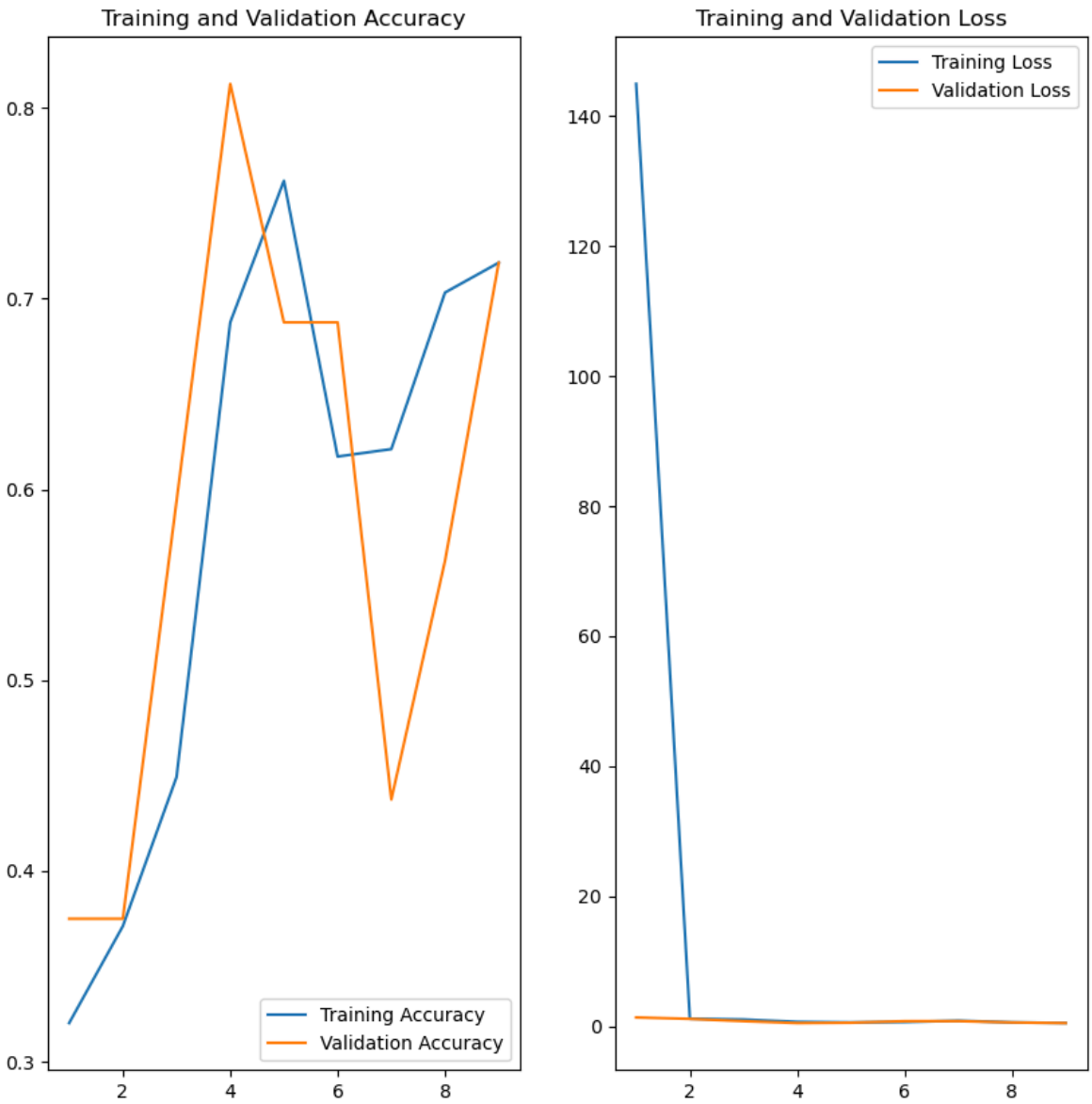
```
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=[early_stopping]
)

val_loss, val_accuracy = model.evaluate(val_ds)
print(f"Validation Loss: {val_loss:.4f}")
print(f"Validation Accuracy: {val_accuracy:.4f}")
```

```
Class names: ['BlackBerry', 'Blueberry', 'strawberry']
Number of classes: 3
Unique labels in train dataset: [0 2 1]
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 8s 710ms/step - accuracy: 0.3066 - loss: 233.3808 - val_
accuracy: 0.3750 - val_loss: 1.3667
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 584ms/step - accuracy: 0.3435 - loss: 1.2603 - val_ac
curacy: 0.3750 - val_loss: 1.1420
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 583ms/step - accuracy: 0.4102 - loss: 1.0865 - val_ac
curacy: 0.5938 - val_loss: 0.7912
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 549ms/step - accuracy: 0.6739 - loss: 0.7244 - val_ac
curacy: 0.8125 - val_loss: 0.5030
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 545ms/step - accuracy: 0.7895 - loss: 0.5318 - val_ac
curacy: 0.6875 - val_loss: 0.5578
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 538ms/step - accuracy: 0.6380 - loss: 0.6781 - val_ac
curacy: 0.6875 - val_loss: 0.7847
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 543ms/step - accuracy: 0.6908 - loss: 0.8348 - val_ac
curacy: 0.4375 - val_loss: 0.7987
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 548ms/step - accuracy: 0.7303 - loss: 0.5937 - val_ac
curacy: 0.5625 - val_loss: 0.5955
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 548ms/step - accuracy: 0.7073 - loss: 0.5451 - val_ac
curacy: 0.7188 - val_loss: 0.5002
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step - accuracy: 0.8125 - loss: 0.5030
Validation Loss: 0.5030
Validation Accuracy: 0.8125
```

In [125…

```
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 Accura
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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

```
plt.show()
```



In [126…

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

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `ker
as.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 `ke
ras.saving.save_model(model, 'my_model.keras')`.
```

In [138…

```
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'D:\ATMA\sem 5\Mesin\BestModel_AlexNet_Tensorflow.h5')
class_names = ['blakcberry', 'blueberry', 'strawberry']

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 {confidenc
    except Exception as e:
        return f"Terjadi kesalahan: {e}"

result = classify_images(r'D:\ATMA\sem 5\Mesin\UAS\test_data\strawberry\strawber
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be b
uilt. `model.compile_metrics` will be empty until you train or evaluate the mode
l.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 117ms/step
Prediksi: strawberry
Confidence: 57.56%
Prediksi: strawberry dengan confidence 57.56%. Gambar asli disimpan di predicted_
image.jpg.
```

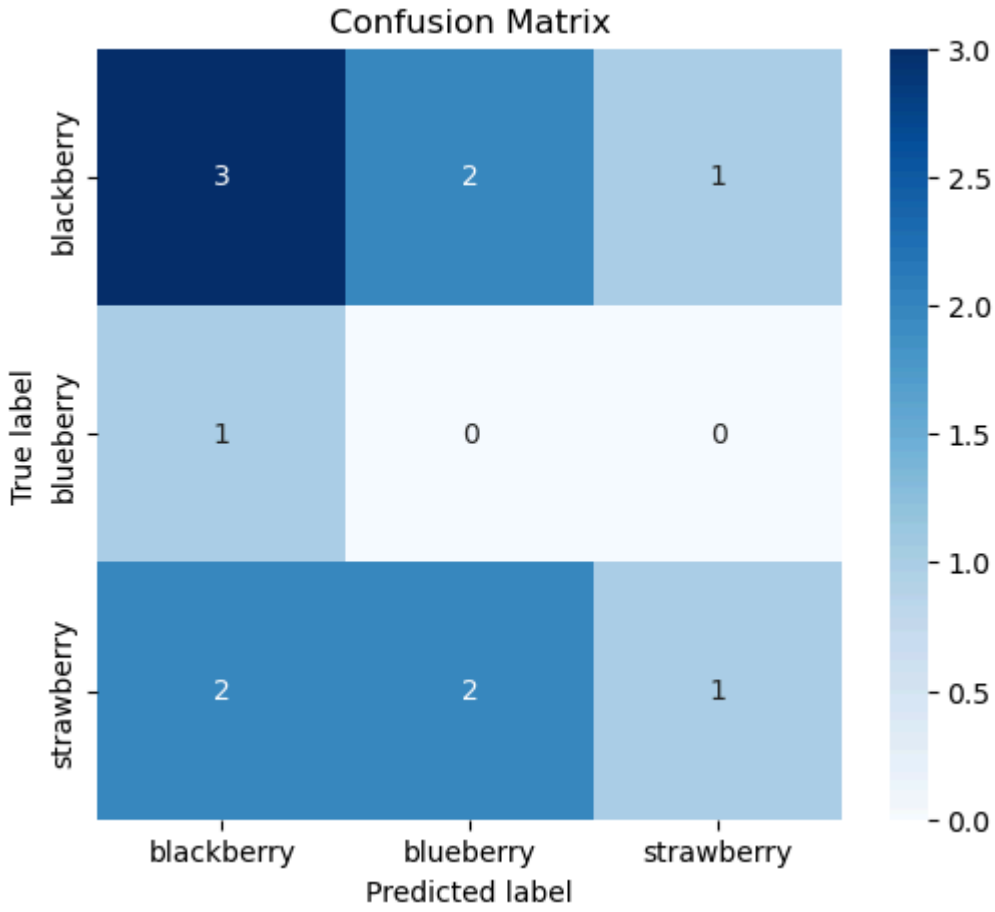
In [130…

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

model = load_model(r'D:\ATMA\sem 5\Mesin\BestModel_AlexNet_Tensorflow.h5')

test_data = test_ds

class_names = ['blackberry', 'blueberry', 'strawberry']

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

true_labels = []  # Store true labels as integers
for _, labels in test_data:
    true_labels.extend(labels.numpy())

# Confusion Matrix
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

# Hitung metrik kinerja
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=class_names, yticklabels=class_names)
plt.title('Confusion Matrix')
```

```
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

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

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 501ms/step
```




```
Confusion Matrix:
 [[3 2 1]
 [1 0 0]
 [2 2 1]]
Akurasi: 0.3333333333333333
Presisi: [0.5 0.  0.5]
Recall: [0.5 0.  0.2]
F1 Score: [0.5               nan 0.28571429]
```

Andreano Yong / 220711682 / GoogleNet

In [99]:
```
import tensorflow as tf

import numpy as np
from matplotlib import pyplot as plt

data_dir = r"C:\train_data_uas"

data = tf.keras.utils.image_dataset_from_directory(data_dir,seed = 123, image_si
print(data.class_names)

class_names = data.class_names
```

```
Found 300 files belonging to 3 classes.
['BlackBerry', 'Blueberry', 'strawberry']
```

In [100…
```
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.
```

In [101…
```
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)
```

```
Total Images: 10
Train Images: 9
Validation Images: 1
```

In [102…
```
import matplotlib.pyplot as plt

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


for images, labels in data.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')
```

In [103…
```
for images, labels in data.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

```
(16, 180, 180, 3)
```

In [104…
```
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model

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


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

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

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

In [105…

```
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()
```

```
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 (MaxPooling2D)	(None, 22, 22, 192)	0	conv2d_2[0][0]
conv2d_4 (Conv2D)	(None, 22, 22, 96)	18,528	max_pooling2d_1[…
conv2d_6 (Conv2D)	(None, 22, 22, 16)	3,088	max_pooling2d_1[…
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 192)	0	max_pooling2d_1[…
conv2d_3 (Conv2D)	(None, 22, 22, 64)	12,352	max_pooling2d_1[…
conv2d_5 (Conv2D)	(None, 22, 22, 128)	110,720	conv2d_4[0][0]
conv2d_7 (Conv2D)	(None, 22, 22, 32)	12,832	conv2d_6[0][0]
conv2d_8 (Conv2D)	(None, 22, 22, 32)	6,176	max_pooling2d_2[…
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,	221,376	conv2d_10[0][0]

	192)		
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 (Concatenate)	(None, 22, 22, 480)	0	conv2d_9[0][0], conv2d_11[0][0], conv2d_13[0][0], conv2d_14[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 11, 11, 480)	0	concatenate_1[0]…
conv2d_16 (Conv2D)	(None, 11, 11, 96)	46,176	max_pooling2d_4[…
conv2d_18 (Conv2D)	(None, 11, 11, 16)	7,696	max_pooling2d_4[…
max_pooling2d_5 (MaxPooling2D)	(None, 11, 11, 480)	0	max_pooling2d_4[…
conv2d_15 (Conv2D)	(None, 11, 11, 192)	92,352	max_pooling2d_4[…
conv2d_17 (Conv2D)	(None, 11, 11, 208)	179,920	conv2d_16[0][0]
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, 64)	38,464	conv2d_24[0][0]

conv2d_26 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_6[…
concatenate_3 (Concatenate)	(None, 11, 11, 512)	0	conv2d_21[0][0], conv2d_23[0][0], conv2d_25[0][0], conv2d_26[0][0]
conv2d_28 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]…
conv2d_30 (Conv2D)	(None, 11, 11, 24)	12,312	concatenate_3[0]…
max_pooling2d_7 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_3[0]…
conv2d_27 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]…
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, 288)	373,536	conv2d_34[0][0]
conv2d_37 (Conv2D)	(None, 11, 11, 64)	51,264	conv2d_36[0][0]
conv2d_38 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_8[…
concatenate_5 (Concatenate)	(None, 11, 11, 528)	0	conv2d_33[0][0], conv2d_35[0][0], conv2d_37[0][0], conv2d_38[0][0]

conv2d_40 (Conv2D)	(None, 11, 11, 160)	84,640	concatenate_5[0]…
conv2d_42 (Conv2D)	(None, 11, 11, 32)	16,928	concatenate_5[0]…
max_pooling2d_9 (MaxPooling2D)	(None, 11, 11, 528)	0	concatenate_5[0]…
conv2d_39 (Conv2D)	(None, 11, 11, 256)	135,424	concatenate_5[0]…
conv2d_41 (Conv2D)	(None, 11, 11, 320)	461,120	conv2d_40[0][0]
conv2d_43 (Conv2D)	(None, 11, 11, 128)	102,528	conv2d_42[0][0]
conv2d_44 (Conv2D)	(None, 11, 11, 128)	67,712	max_pooling2d_9[…
concatenate_6 (Concatenate)	(None, 11, 11, 832)	0	conv2d_39[0][0], conv2d_41[0][0], conv2d_43[0][0], conv2d_44[0][0]
max_pooling2d_10 (MaxPooling2D)	(None, 6, 6, 832)	0	concatenate_6[0]…
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 (MaxPooling2D)	(None, 6, 6, 832)	0	max_pooling2d_10…
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 (Concatenate)	(None, 6, 6, 832)	0	conv2d_45[0][0], 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 (MaxPooling2D)	(None, 6, 6, 832)	0	concatenate_7[0]…
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 (Concatenate)	(None, 6, 6, 1024)	0	conv2d_51[0][0], conv2d_53[0][0], conv2d_55[0][0], conv2d_56[0][0]
average_pooling2d (AveragePooling2D)	(None, 4, 4, 1024)	0	concatenate_8[0]…
dropout (Dropout)	(None, 4, 4, 1024)	0	average_pooling2…
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)

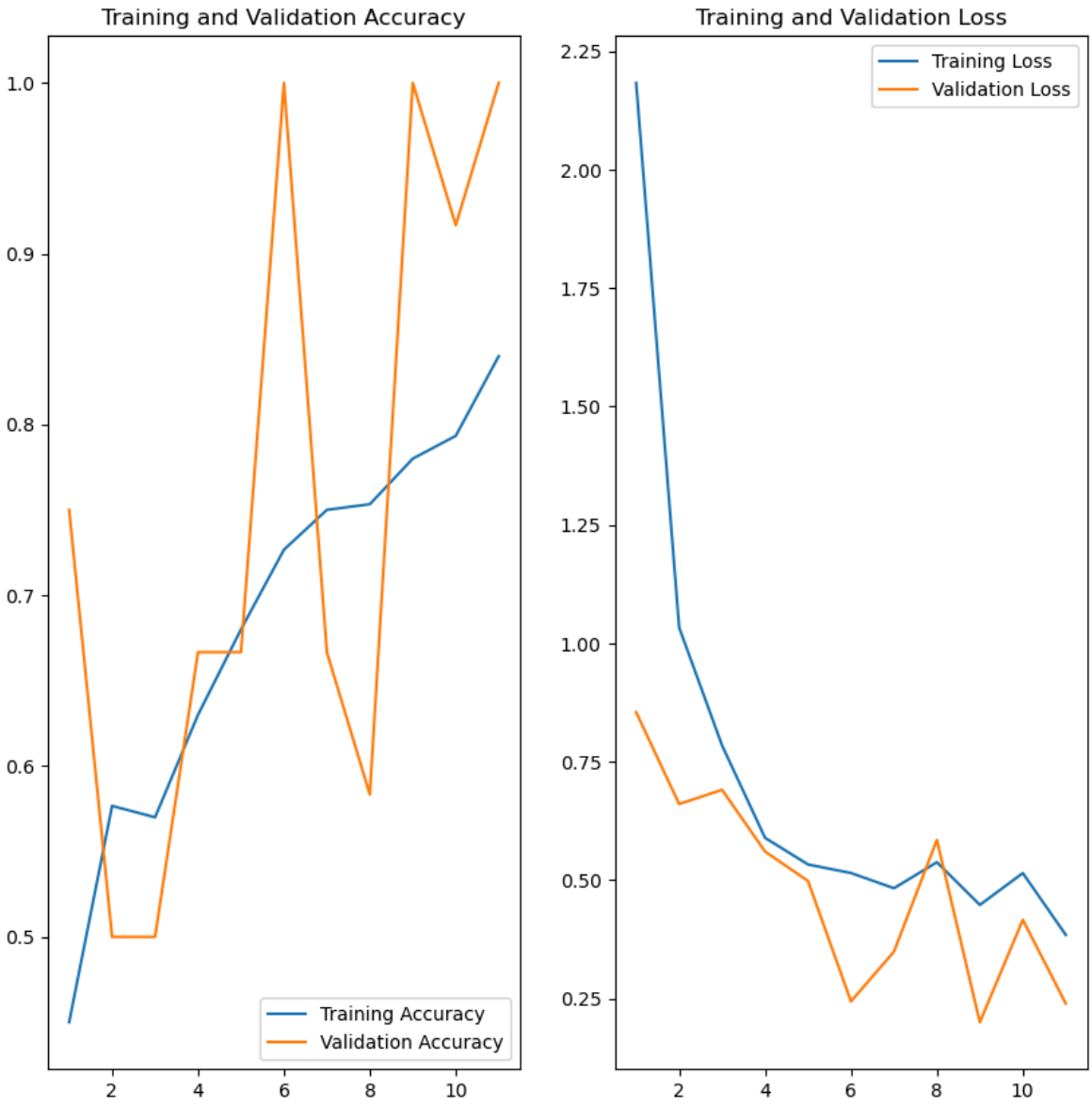
In [106…

```
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
19/19 ━━━━━━━━━━━━━━━━━━━━ 32s 517ms/step - accuracy: 0.3718 - loss: 3.9434 - val
_accuracy: 0.7500 - val_loss: 0.8545
Epoch 2/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 8s 443ms/step - accuracy: 0.5541 - loss: 1.2925 - val_
accuracy: 0.5000 - val_loss: 0.6609
Epoch 3/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 9s 453ms/step - accuracy: 0.5584 - loss: 0.8697 - val_
accuracy: 0.5000 - val_loss: 0.6908
Epoch 4/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 9s 488ms/step - accuracy: 0.6372 - loss: 0.5873 - val_
accuracy: 0.6667 - val_loss: 0.5606
Epoch 5/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 10s 505ms/step - accuracy: 0.6697 - loss: 0.5532 - val
_accuracy: 0.6667 - val_loss: 0.4984
Epoch 6/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 9s 454ms/step - accuracy: 0.7156 - loss: 0.5191 - val_
accuracy: 1.0000 - val_loss: 0.2442
Epoch 7/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 8s 431ms/step - accuracy: 0.7473 - loss: 0.4696 - val_
accuracy: 0.6667 - val_loss: 0.3492
Epoch 8/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 8s 416ms/step - accuracy: 0.7235 - loss: 0.5249 - val_
accuracy: 0.5833 - val_loss: 0.5851
Epoch 9/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 8s 427ms/step - accuracy: 0.7681 - loss: 0.4353 - val_
accuracy: 1.0000 - val_loss: 0.2002
Epoch 10/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 8s 420ms/step - accuracy: 0.7768 - loss: 0.5679 - val_
accuracy: 0.9167 - val_loss: 0.4165
Epoch 11/30
19/19 ━━━━━━━━━━━━━━━━━━━━ 8s 432ms/step - accuracy: 0.8622 - loss: 0.4059 - val_
accuracy: 1.0000 - val_loss: 0.2397
```

In [107…

```
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 Accura
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()
```



In [116…
```
model.save('BestModel_GoogleNet_Tensorflow.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `ker
as.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 `ke
ras.saving.save_model(model, 'my_model.keras')`.
```

In [118…
```
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:\UAS_Bagian_GoogleNet\BestModel_GoogleNet_Tensorflow.h5')
class_names = ['BlackBerry', 'Blueberry', 'strawberry']


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 {confidenc
    except Exception as e:
        return f"Terjadi kesalahan: {e}"


result = classify_images(r'C:\test_data_uas\Blueberry\625bcc28ebdb6b529bc8e680d8
print(result)
```

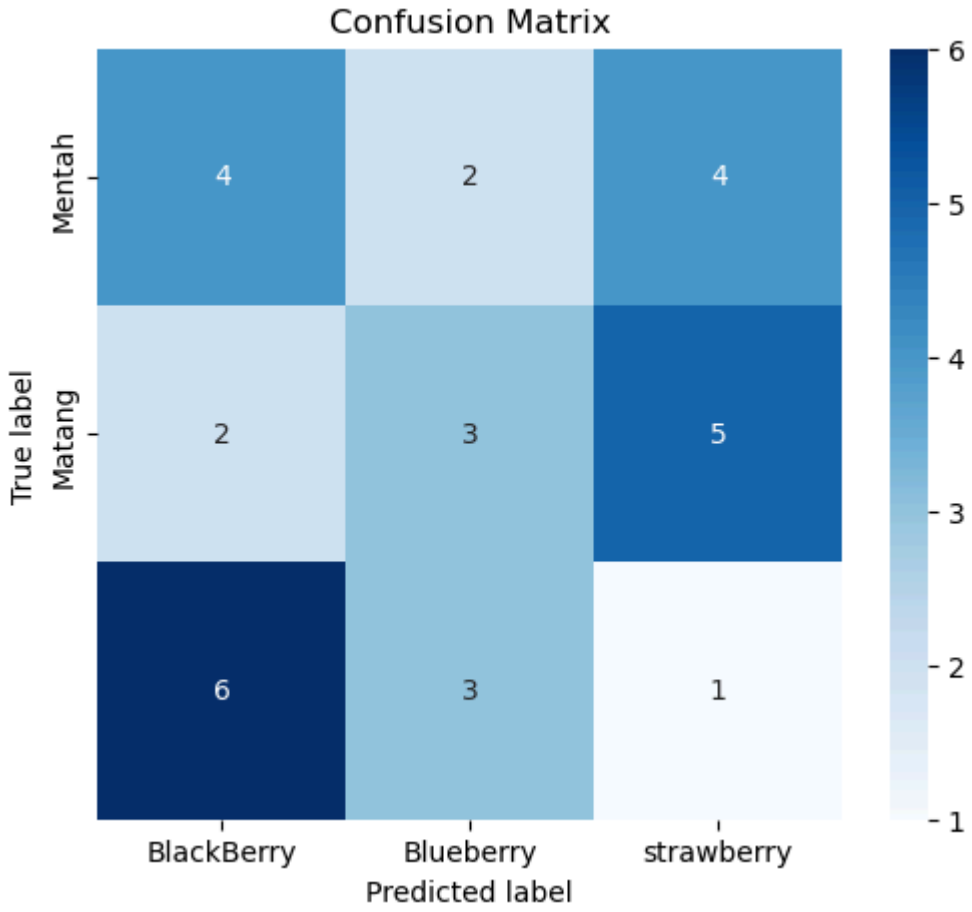
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be b
uilt. `model.compile_metrics` will be empty until you train or evaluate the mode
l.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 835ms/step
Prediksi: Blueberry
Confidence: 50.43%
Prediksi: Blueberry dengan confidence 50.43%. Gambar asli disimpan di bluee.jpg.
```

In [119…

```
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'C:\test_data_uas',
    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=["BlackBerry", "Blueberry", "strawberry"], yticklabels=[
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 1s/step
```



```
Confusion Matrix:
 [[4 2 4]
 [2 3 5]
 [6 3 1]]
Akurasi: 0.26666666666666666
Presisi: [0.33333333 0.375      0.1       ]
Recall: [0.4 0.3 0.1]
F1 Score: [0.36363636 0.33333333 0.1       ]
```

Stewart Sidauruk - 220711816 - TensorFlow - Cabai - VGG-16

In [49]:
```
# Import necessary libraries
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.applications import VGG16
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
from tensorflow.keras.preprocessing.image import load_img, ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.callbacks import EarlyStopping
```

In [50]:
```
count = 0
dirs = os.listdir(r'C:\Users\stewa\Downloads\UAS\train_data')
for dir in dirs:
    files = list(os.listdir(r'C:\Users\stewa\Downloads\UAS\train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
BlackBerry Folder has 100 Images
Blueberry Folder has 100 Images
strawberry Folder has 100 Images
Images Folder has 300 Images
```

In [51]:
```
import os
from PIL import Image

base_dir = r'C:\Users\stewa\Downloads\UAS\train_data'

for root, _, files in os.walk(base_dir):
    for file in files:
        file_path = os.path.join(root, file)
        try:
            img = Image.open(file_path)
            img.verify()
        except Exception as e:
            print(f"Corrupted or unsupported file: {file_path} - {e}")
```

In [52]:
```
# Parameter
base_dir = r'C:\Users\stewa\Downloads\UAS\train_data'
img_size = (224, 224)
batch = 32
validation_split = 0.1
test_split = 0.1
```

In [53]:
```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=img_size,
```

```
    batch_size=batch,
)
```

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

In [54]:
```
class_names = dataset.class_names
print("Class Names:", class_names)
```

```
Class Names: ['BlackBerry', 'Blueberry', 'strawberry']
```

In [55]:
```
total_count = len(dataset)
val_count = int(total_count * validation_split)
tes_count = int(total_count * test_split)
train_count = total_count - val_count - tes_count

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

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

In [56]:
```
train_ds = dataset.take(train_count)
remaining = dataset.skip(train_count)
val_ds = remaining.take(val_count)
tes_ds = remaining.skip(tes_count)
```

In [57]:
```
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')
```

In [58]:
```
import numpy as np

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

```
(32, 224, 224, 3)
```

In [59]:
```
AUTOTUNE = tf.data.AUTOTUNE
```

In [60]:
```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

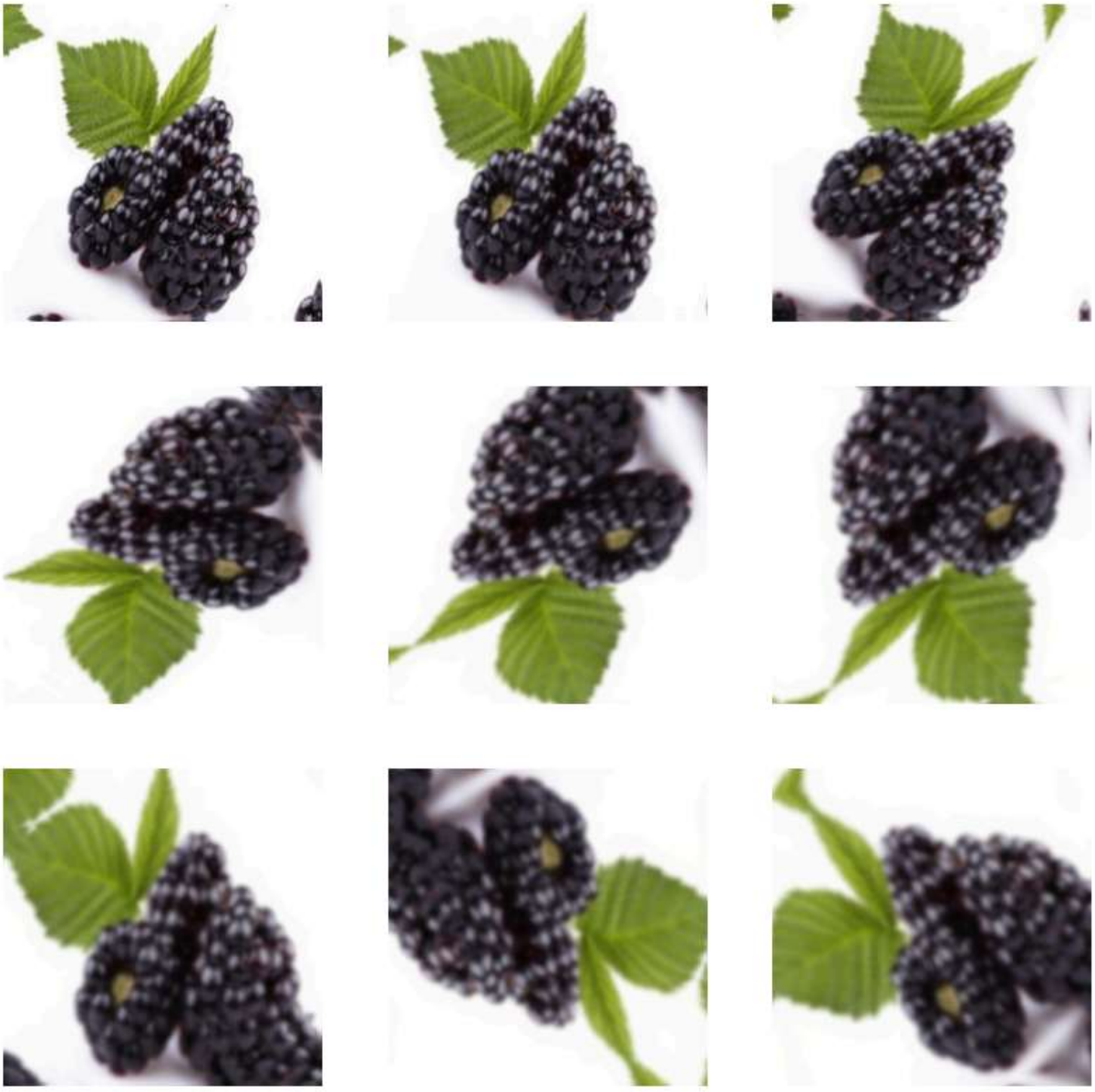
In [61]:
```
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

In [62]:
```
data_augmentation = Sequential([
    layers.RandomFlip("horizontal_and_vertical", input_shape=(img_size[0], img_size
    layers.RandomRotation(0.1),
```

```
    layers.RandomZoom(0.1)
])
```

```
c:\Users\stewa\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)
```

In [63]:
```
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')
```



In [64]:
```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras import layers

base_model = VGG16(include_top=False, input_shape=(img_size[0], img_size[1], 3))

base_model.trainable = True
fine_tune_at = len(base_model.layers) // 2
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(len(class_names), activation='softmax')
])

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

In [65]:
```
from tensorflow.keras.optimizers import Adam

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

In [66]:
```
model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
sequential_4 (Sequential)	(None, 224, 224, 3)	0
rescaling_2 (Rescaling)	(None, 224, 224, 3)	0
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 512)	0
dense_4 (Dense)	(None, 128)	65,664
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 3)	387

Total params: 14,780,739 (56.38 MB)

Trainable params: 13,635,331 (52.01 MB)

Non-trainable params: 1,145,408 (4.37 MB)

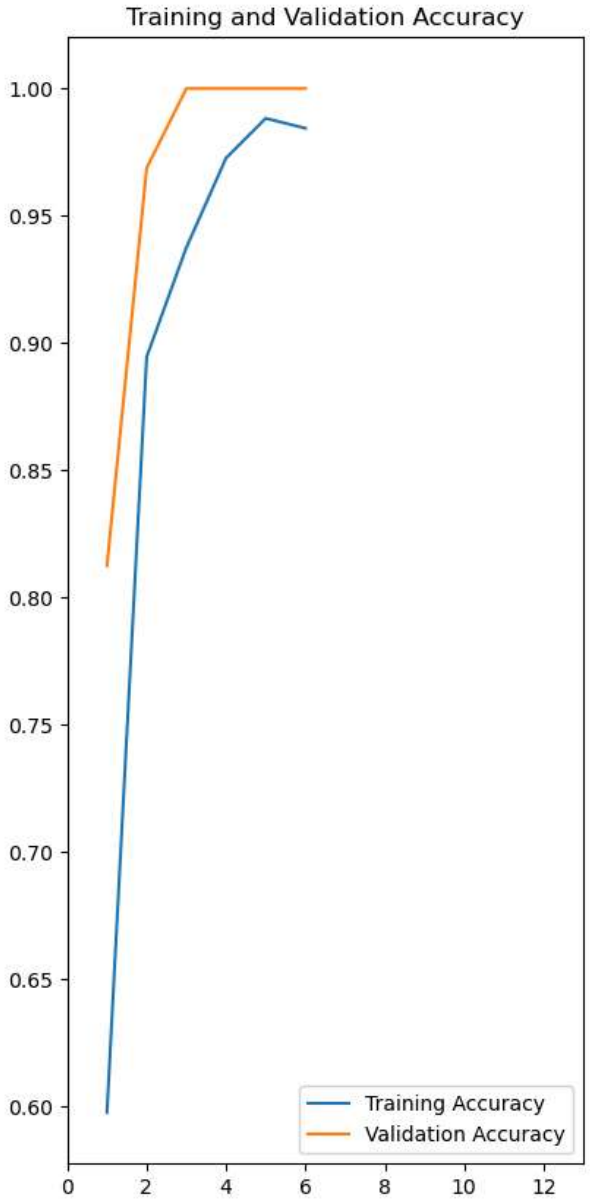
In [67]:
```
from tensorflow.keras.callbacks import EarlyStopping

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

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

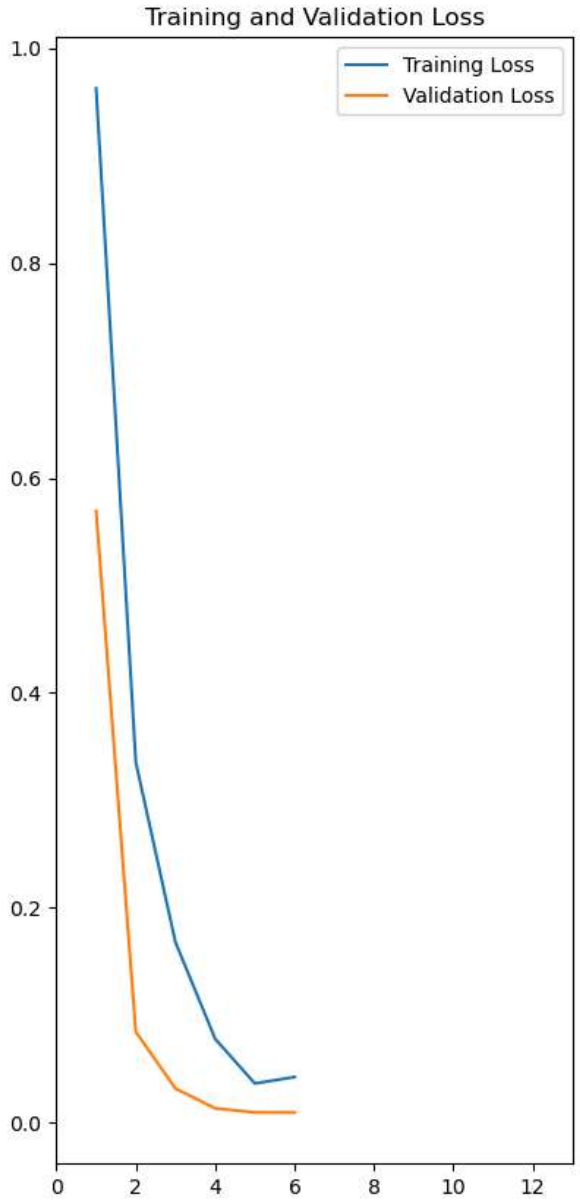
```
Epoch 1/30
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 34s 4s/step - accuracy: 0.5163 - loss: 1.0746 - val_accurac
y: 0.8125 - val_loss: 0.5692
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 30s 4s/step - accuracy: 0.8526 - loss: 0.4265 - val_accurac
y: 0.9688 - val_loss: 0.0848
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 31s 4s/step - accuracy: 0.9518 - loss: 0.1574 - val_accurac
y: 1.0000 - val_loss: 0.0320
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 30s 4s/step - accuracy: 0.9752 - loss: 0.0795 - val_accurac
y: 1.0000 - val_loss: 0.0136
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 31s 4s/step - accuracy: 0.9840 - loss: 0.0499 - val_accurac
y: 1.0000 - val_loss: 0.0098
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 30s 4s/step - accuracy: 0.9872 - loss: 0.0412 - val_accurac
y: 1.0000 - val_loss: 0.0098
```

In [68]:

```
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()
```





In [69]:
```
#menyimpan model yang telah dilatih
model.save('BestModel_VGG-16_Tensorflow.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')`.
```

In [70]:
```
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\stewa\Downloads\TUBES_ML\BestModel_VGG-16_Tensorflow.
class_names = ['BlackBerry', 'Blueberry', 'strawberry']

def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
```

```
        input_image = tf.keras.utils.load_img(image_path, target_size=(224, 224))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)

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

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

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

        return f"Prediksi: {class_names[class_idx]} dengan confidence {confidence:.
    except Exception as e:
        return f"Terjadi kesalahan: {e}"

result = classify_images(r'C:\Users\stewa\Downloads\UAS\test_data\blackberry\Home.j
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be buil
t. `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_pre
dict_function.<locals>.one_step_on_data_distributed at 0x000001B735547B00> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings cou
ld be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors wit
h different shapes, (3) passing Python objects instead of tensors. For (1), please d
efine your @tf.function outside of the loop. For (2), @tf.function has reduce_retrac
ing=True option that can avoid unnecessary retracing. For (3), please refer to http
s://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorfl
ow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:6 out of the last 6 calls to <function TensorFlowTrainer.make_pre
dict_function.<locals>.one_step_on_data_distributed at 0x000001B735547B00> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings cou
ld be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors wit
h different shapes, (3) passing Python objects instead of tensors. For (1), please d
efine your @tf.function outside of the loop. For (2), @tf.function has reduce_retrac
ing=True option that can avoid unnecessary retracing. For (3), please refer to http
s://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorfl
ow.org/api_docs/python/tf/function for  more details.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 212ms/step
Prediksi: BlackBerry
Confidence: 57.51%
Prediksi: BlackBerry dengan confidence 57.51%. Gambar asli disimpan di blackberry.jp
g.
```

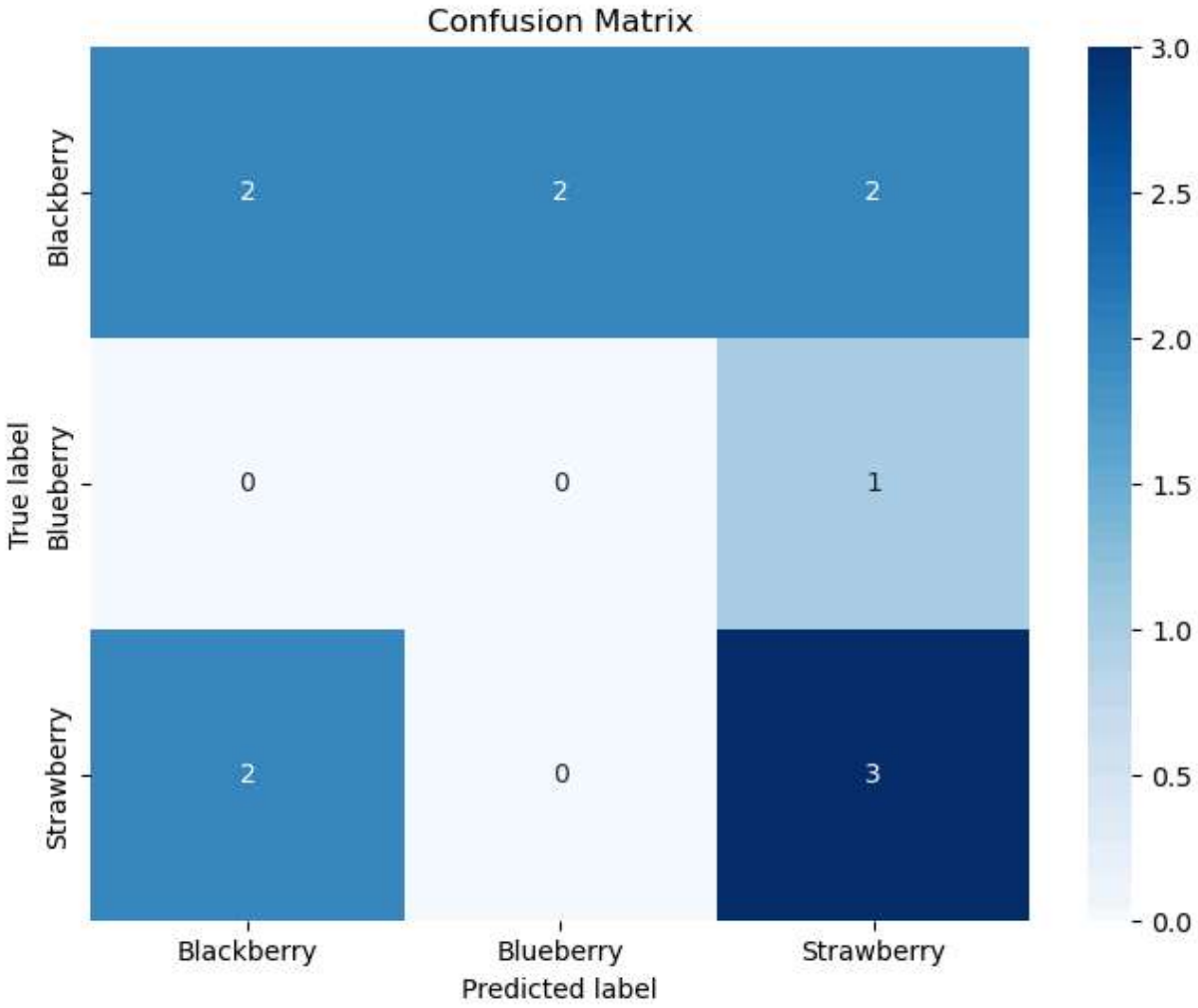
In [71]:

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

vgg16_model = load_model(r'C:\Users\stewa\Downloads\TUBES_ML\BestModel_VGG-16_Tenso
```

```
# Memuat data uji yang sebenarnya
test_data = tes_ds

# Melakukan prediksi menggunakan model
y_pred = vgg16_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
# Extract actual labels from test_data
true_labels = []  # Store true labels as integers
for _, labels in test_data:
    true_labels.extend(labels.numpy())  # Directly convert TensorFlow tensor to Num

true_labels = tf.convert_to_tensor(true_labels)  # Convert the list to a TensorFlow
  # Mengonversi list ke tensor untuk perhitungan

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

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

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

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

# Visualisasi Confusion Matrix
plt.figure(figsize=(8, 6))  # Mengatur ukuran gambar
sns.heatmap(
    conf_mat.numpy(),
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=["Blackberry", "Blueberry", "Strawberry"],  # Label prediksi
    yticklabels=["Blackberry", "Blueberry", "Strawberry"],  # Label asli
)
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Menampilkan 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 buil
t. `model.compile_metrics` will be empty until you train or evaluate the model.
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
```




```
Confusion Matrix:
 [[2 2 2]
 [0 0 1]
 [2 0 3]]
Akurasi: 0.4166666666666667
Presisi: [0.5 0.  0.5]
Recall: [0.33333333 0.         0.6       ]
F1 Score: [0.4              nan 0.54545455]
```

Octavionus Samuel Kusuma Wardana, 220711661, TensorFlow, topik studi kasus, MobileNet

MobileNet

Import Library

- Tahap pertama adalah import seluruh library yang dibutuhkan

In [1]:
```
#Import library
import os
import numpy as np

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

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

Load Data

- Load dataset berdasarkan path dimana dataset disimpan

In [2]:
```
count = 0 #digunakan untuk menghitung jumlah gambar
dirs = os.listdir(r'C:\Users\stewa\Downloads\UAS\train_data')
for dir in dirs:
    files = list(os.listdir(r'C:\Users\stewa\Downloads\UAS\train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
BlackBerry Folder has 100 Images
Blueberry Folder has 100 Images
strawberry Folder has 100 Images
Images Folder has 300 Images
```

Load Images into Arrays as Dataset

- Membuat dataset dari gambar yang ada di direktori

In [3]:
```
# # Parameter
# base_dir = r'train_data_2' #direktori folder dataset
# img_size = 180 #mengubah ukuran gambar menjadi 180
# batch = 32 #jumlah sample (gambar) yang akan diproses pada satu kali iterasi
# validation_split = 0.1 #data pelatihan yang akan digunakan sebagai data validasi
```

In [4]:
```
# Parameter
base_dir = r'C:\Users\stewa\Downloads\UAS\train_data' #direktori folder dataset
img_size = 180 #mengubah ukuran gambar menjadi 180
batch = 1 #jumlah sample (gambar) yang akan diproses pada satu kali iterasi
validation_split = 0.1 #data pelatihan yang akan digunakan sebagai data validasi
```

- Memasukkan parameter yang telah di definisikan tadi untuk membuat dataset dari gambar di direktori

In [5]:
```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir, #path direktori, subfolder dianggap sebagai label
    seed=123, #untuk memastikan proses pemisahan data selalu konsisten (random_stat
    image_size=(img_size, img_size), #ukuran gambar diubah (resize) menjadi 180x180
    batch_size=batch, #jumlah gambar yang akan dikelompokkan
)
```

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

In [6]:
```
#mendapatkan nama kelas dari dataset
class_names = dataset.class_names #dataset.class_names akan mengambil daftar nama k
print("Class Names:", class_names)
```

```
Class Names: ['BlackBerry', 'Blueberry', 'strawberry']
```

Train-Validation-Test Split

- Membagi dataset menjadi tiga subset yaitu train, validation, dan test
  - Train, digunakan untuk melatih model agar mengenali pola dalam data
  - Validation, digunakan untuk mengevaluasi performa model selama pelatihan
  - Test, digunakan untuk menguji model setelah pelatihan

In [7]:
```
# ###Terdapat code yang hilang disini! lihat modul untuk menemukanya menghitung jum

# 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)
```

In [8]:
```
###Terdapat code yang hilang disini! lihat modul untuk menemukanya menghitung jumla

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)
```

```
Total Images: 300
Train Images: 270
Validation Images: 30
```

In [9]:
```
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
#Cell ini untuk membagi dataset bang

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

In [10]:
```
for images, labels in train_ds.take(1):  # Ambil 1 batch untuk diperiksa
    print("Shape of images:", images.shape)
    print("Shape of labels:", labels.shape)

    images_array = np.array(images)
    print(images_array.shape)
```

```
Shape of images: (1, 180, 180, 3)
Shape of labels: (1,)
(1, 180, 180, 3)
```

In [11]:
```
import matplotlib.pyplot as plt

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

# ###Terdapat code yang hilang disini! lihat modul untuk menemukanya
# for images, labels in train_ds.take(1):
#     for i in range(9): #mengambil 1 batch pertama dari train_ds
#         plt.subplot(3,3, i+1) #menyiapkan subplot dengan grid 3x3 dan menempatkan
#         plt.imshow(images[i].numpy().astype('uint8')) #menampilkan gambar dan men
#         plt.title(class_names[labels[i]]) #menampilkan judul gambar sesuai dengan
#         plt.axis('off') #menonaktifkan sumbu pada gambar agar tidak terlihat
```

In [12]:
```
import numpy as np

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

```
    #180: Tinggi gambar dalam piksel
    #3: Jumlah channel gambar (RGB)
```

```
(1, 180, 180, 3)
```

In [13]:
```
#Mengatur AUTOTUNE untuk pemrosesan data otomatis oleh tensorflow
#AUTOTUNE digunakan untuk memungkinkan tensorflow mengoptimalkan jumlah thread seca
AUTOTUNE = tf.data.AUTOTUNE
```

In [14]:
```
#mengoptimalkan dataset pelatihan (train_ds)
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
#cache digunakan untuk menyimpan dataser di memori agar lebih cepat diakses
#shuffle mengacak data dalam batch agar model tidak terlalu terlatih pada urutan te
#prefetch untuk menyiapkan data batch berikutnya secara otomatis
```

In [15]:
```
#mengoptimalkan dataset validasi (val_ds)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

Data Augmentation

- Digunakan untuk menambah variasi data pelatihan dengan membuat gambar baru dari yang sudah ada seperti dengan rotasi, flipping, zooming, dan sebagainy
- Untuk mengurangi overfitting dan memperbesar dataset tanpa mengumpulkan data baru

In [16]:
```
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape = (img_size,img_size,3)), #membalik
    layers.RandomRotation(0.1), #merotasi gambar secara acak dalam kisaran 0°-36° (
    layers.RandomZoom(0.1) #melakukan zoom in/zoom out secara acak dengan rentang 1
])
```

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

In [17]:
```
# #sama seperti sebelumnya, code ini digunakan untuk menampilkan gambar dari data_a
# 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')
```

MobileNet

- Salah satu algoritma yang dirancang untuk perangkat dengan keterbatasan sumber daya seperti smartphone

In [18]:
```
# #import library yang dibutuhkan
# from tensorflow.keras.applications import MobileNet #digunakan untuk memanfaatkan
# from tensorflow.keras.models import Model #digunakan untuk membuat dan mengonfigu

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

# #membuka (unfreeze beberapa lapisan untuk proses fine tuning)
# base_model.trainable = True #seluruh model bisa dilatih
# fine_tune_at = len(base_model.layers) // 2  #menentukan bahwa setengah lapisan te
# for layer in base_model.layers[:fine_tune_at]:
#     layer.trainable = False #mengunci (freeze) lapisan pertama hingga setengah ba

# #membuat model akhir dengan lapisan tambahan ////////////////////////////////////
# ###Terdapat code yang hilang disini! lihat modul untuk menemukanya

# model = Sequential([
#     data_augmentation, #data augmentasi untuk memperbanyak data latih
#     layers.Rescaling(1./255), #menormalisasi gambar, mengubah nilai pixel dari re
#     base_model, #mobileNet sebagai dasar ekstraksi fitur
#     layers.GlobalAveragePooling2D(), #Lapisan pooling untuk meratakan hasil fitur
#     Dense(128, activation='relu'), #lapisan dense dengan 128 neuron dan aktivasi
#     Dropout(0.3), #dropout dengan 30% neuron yang dihilangkan secara acak selama
#     Dense(len(class_names), activation='softmax') #lapisan output dengan jumlah n
#                                                   #aktivasi softmax untuk klasifi
# ])
```

In [19]:
```
#import library yang dibutuhkan
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten,
from tensorflow.keras import layers

# Membuat model CNN dari awal
model = Sequential([
    data_augmentation,  # Data augmentasi
    layers.Rescaling(1./255, input_shape=(img_size, img_size, 3)),  # Normalisasi

    # Blok pertama: Convolution + Pooling
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    # Blok kedua: Convolution + Pooling
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    # Blok ketiga: Convolution + Pooling
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
```

```
    # Global pooling
    layers.GlobalAveragePooling2D(),

    # Fully connected layer
    Dense(128, activation='relu'),
    Dropout(0.3),  # Dropout

    # Output layer
    Dense(len(class_names), activation='softmax')  # Output layer
])
```

In [20]:
```
from tensorflow.keras.optimizers import Adam #untuk mengoptimalkan proses pelatihan

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

In [21]:
```
#menampilkan ringkasan dari model
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, 180, 180, 32)	896
max_pooling2d (MaxPooling2D)	(None, 90, 90, 32)	0
conv2d_1 (Conv2D)	(None, 90, 90, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 64)	0
conv2d_2 (Conv2D)	(None, 45, 45, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 128)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 128)	0
dense (Dense)	(None, 128)	16,512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

Total params: 110,147 (430.26 KB)

Trainable params: 110,147 (430.26 KB)

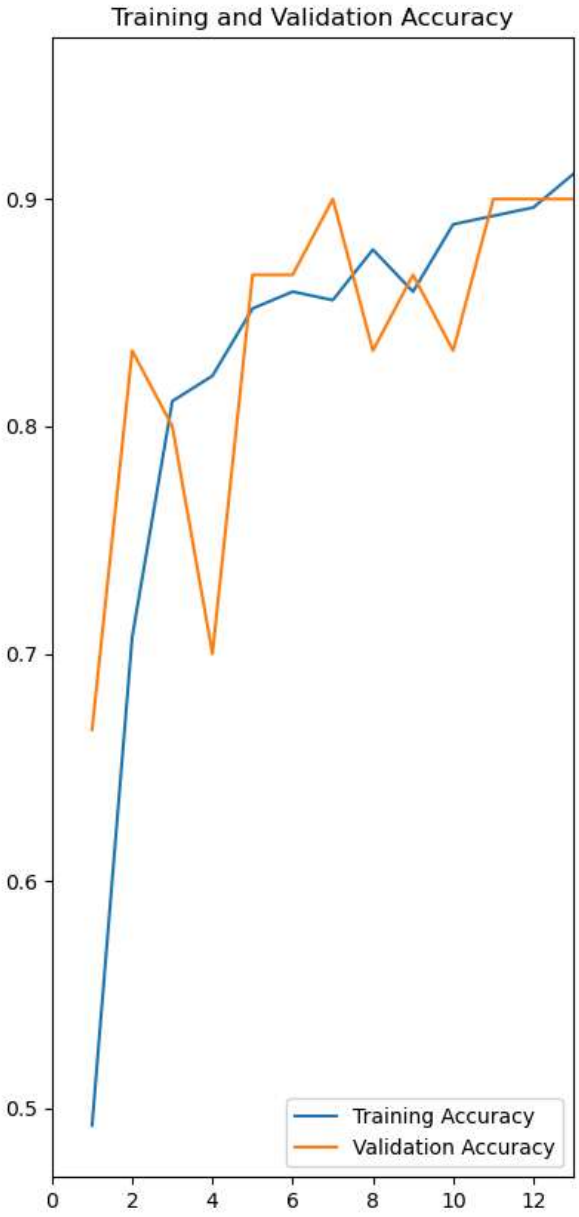
Non-trainable params: 0 (0.00 B)

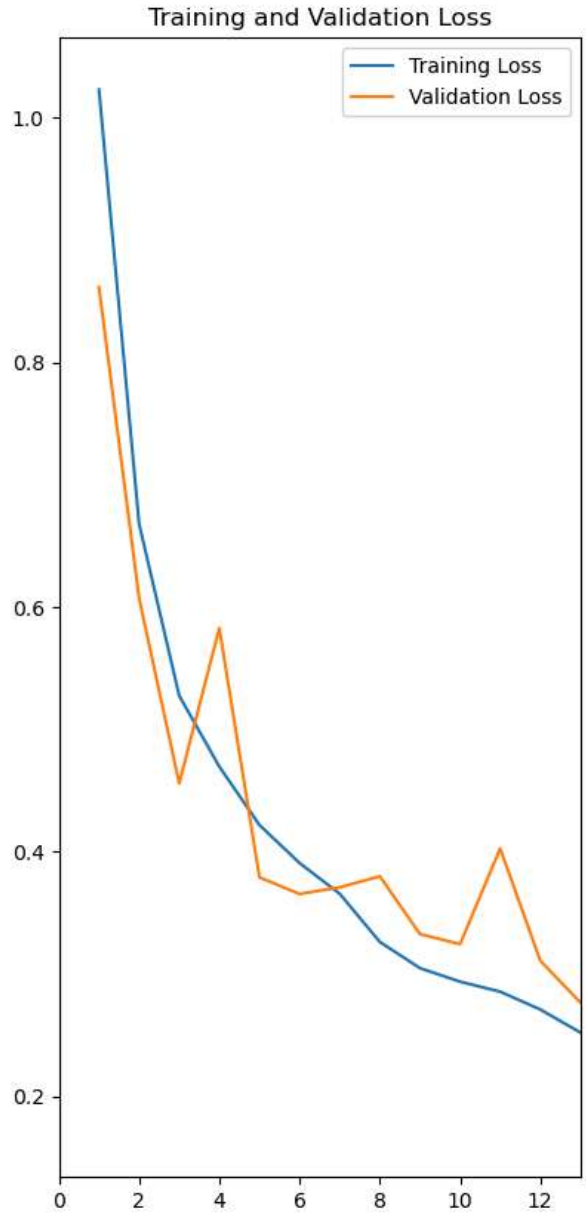
In [22]:
```
#early stopping digunakan untuk menghentikan pelatihan lebih awal jika model tidak
from tensorflow.keras.callbacks import EarlyStopping

#Ada fungsi early stopping disini, jangan keskip tuan :D
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=7,
                               mode='max')

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

```
Epoch 1/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 6s 16ms/step - accuracy: 0.4009 - loss: 1.0789 - val_ac
curacy: 0.6667 - val_loss: 0.8616
Epoch 2/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.6070 - loss: 0.7803 - val_ac
curacy: 0.8333 - val_loss: 0.6072
Epoch 3/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.8044 - loss: 0.5379 - val_ac
curacy: 0.8000 - val_loss: 0.4557
Epoch 4/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.8229 - loss: 0.4598 - val_ac
curacy: 0.7000 - val_loss: 0.5827
Epoch 5/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 14ms/step - accuracy: 0.8462 - loss: 0.4207 - val_ac
curacy: 0.8667 - val_loss: 0.3789
Epoch 6/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 14ms/step - accuracy: 0.8523 - loss: 0.3821 - val_ac
curacy: 0.8667 - val_loss: 0.3653
Epoch 7/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.8632 - loss: 0.3297 - val_ac
curacy: 0.9000 - val_loss: 0.3706
Epoch 8/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.9070 - loss: 0.3043 - val_ac
curacy: 0.8333 - val_loss: 0.3797
Epoch 9/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.8602 - loss: 0.3139 - val_ac
curacy: 0.8667 - val_loss: 0.3325
Epoch 10/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.9179 - loss: 0.2432 - val_ac
curacy: 0.8333 - val_loss: 0.3242
Epoch 11/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.8962 - loss: 0.2748 - val_ac
curacy: 0.9000 - val_loss: 0.4025
Epoch 12/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.8867 - loss: 0.2795 - val_ac
curacy: 0.9000 - val_loss: 0.3107
Epoch 13/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.9021 - loss: 0.2703 - val_ac
curacy: 0.9000 - val_loss: 0.2765
Epoch 14/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.9061 - loss: 0.2418 - val_ac
curacy: 0.9333 - val_loss: 0.3164
Epoch 15/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.8984 - loss: 0.2235 - val_ac
curacy: 0.9333 - val_loss: 0.3450
Epoch 16/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.9097 - loss: 0.2307 - val_ac
curacy: 0.9333 - val_loss: 0.2951
Epoch 17/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.9299 - loss: 0.1932 - val_ac
curacy: 0.9333 - val_loss: 0.3325
Epoch 18/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.9512 - loss: 0.1828 - val_ac
curacy: 0.9333 - val_loss: 0.3010
Epoch 19/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.9240 - loss: 0.1880 - val_ac
```

```
curacy: 0.9333 - val_loss: 0.3296
Epoch 20/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.9186 - loss: 0.2246 - val_ac
curacy: 0.9333 - val_loss: 0.3760
Epoch 21/30
270/270 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.9290 - loss: 0.1735 - val_ac
curacy: 0.9000 - val_loss: 0.4044
```

In [23]:
```
#membuat range untuk epoch berdasarkan panjang data loss dari pelatihan
ephocs_range = range(1, len(history.history['loss']) + 1)

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

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

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





In [24]:
```
#menyimpan model yang telah dilatih
model.save('model_mobilenet_tubes.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')`.
```

In [25]:
```
model.save('model_mobilenet_tubes.h5', save_format='h5')
```

```
WARNING:absl:The `save_format` argument is deprecated in Keras 3. We recommend removing this argument as it can be inferred from the file path. Received: save_format=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')`.
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')`.
```

In [26]:

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

#memuat model yang sudah dilatih
model = load_model(r'C:\Users\stewa\Downloads\TUBES_ML\model_mobilenet_tubes.h5')
class_names = ['cabai besar', 'cabai keriting','cabai rawit'] #kelas yang ada pada

#fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        #memuat dan mempersiapkan gambar untuk prediksi
        input_image = tf.keras.utils.load_img(image_path, target_size=(180, 180)) #
        input_image_array = tf.keras.utils.img_to_array(input_image) #mengubah gamb
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)  #menambahkan di
                                                                    #dimensi menjad

        #melakukan prediksi
        predictions = model.predict(input_image_exp_dim) #melakukan prediksi pada g
        result = tf.nn.softmax(predictions[0]) #menghitung hasil prediksi menggunak
        class_idx = np.argmax(result) #menemukan indeks kelas dengan probabilitas t
        confidence = np.max(result) * 100 #menghitung confidence dalam persentase

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

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

        return f"Prediksi: {class_names[class_idx]} dengan confidence {confidence:.
    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'test_data/Keriting/Keriting-merah.jpg', save_path='matan
# result = classify_images(r'test_data/Matang/Mature_Dragon_Original_Data0020.jpg',
# test_data/Keriting/cabe-keriting-hijau-1-SESA_1-removebg-preview.jpg
# test_data/Keriting/cabe-keriting-hijau-1-SESA_1-removebg-preview.webp
```

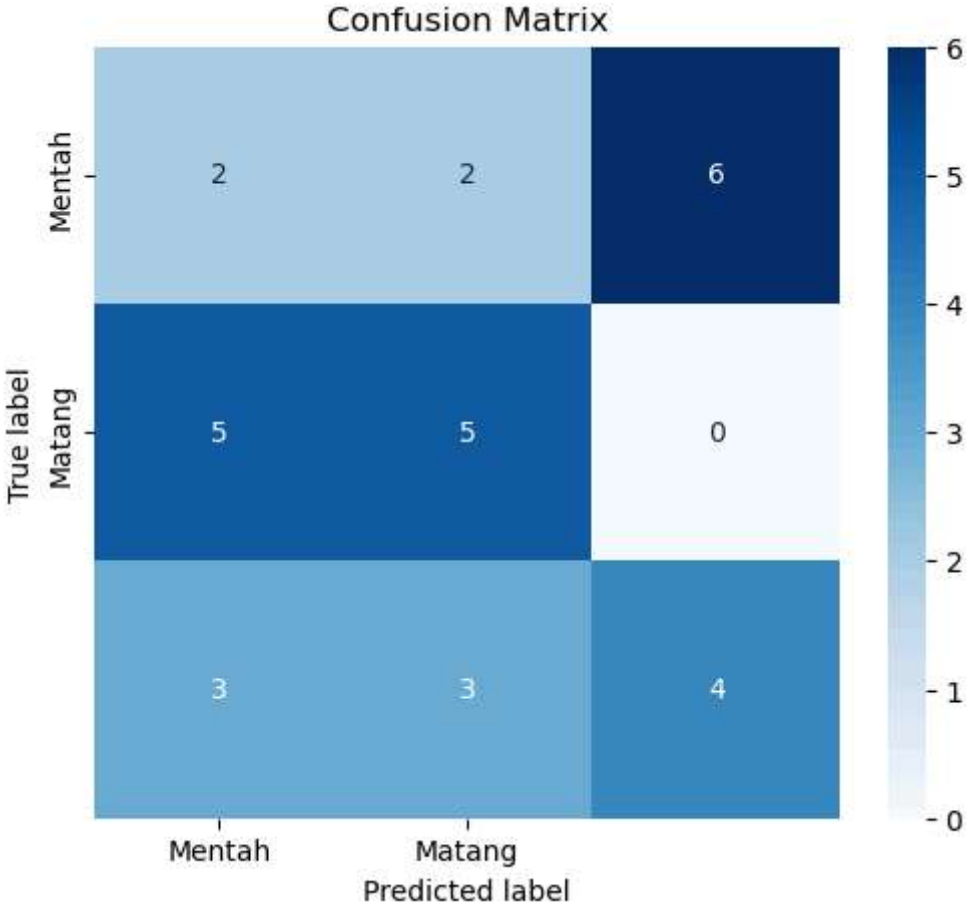
```
# C:\Users\Asus\Kuliah\Semester 5\PMDPM (Pembelajaran Mesin dan Pembelajaran Mendal
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be buil
t. `model.compile_metrics` will be empty until you train or evaluate the model.
Terjadi kesalahan: [Errno 2] No such file or directory: 'test_data/Keriting/Keriting
-merah.jpg'
```

In [28]:
```
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

#memuat model yang telah dilatih sebelumnya
mobileNet_model = load_model(r'C:\Users\stewa\Downloads\TUBES_ML\model_mobilenet_tu

#memuat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\stewa\Downloads\UAS\test_data', #direktori data uji
    labels='inferred', #label otomatis dari subfolder yang ada
    label_mode='categorical',  #menghasilkan label dalam bentuk one-hot encoding
    batch_size=32, #ukuran batch untuk pemrosesan
    image_size=(180, 180) #ukuran gambar yang akan diproses
)

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

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

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

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

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

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

#visualisasi Confusion Matrix
plt.figure(figsize=(6, 5)) #mengatur ukuran gambar
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues', #annot=True untuk
                                                                 #fmt='d' untuk men
            xticklabels=["Mentah", "Matang"], yticklabels=["Mentah", "Matang"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
```

```
plt.show()

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

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be buil
t. `model.compile_metrics` will be empty until you train or evaluate the model.
Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 206ms/step
```




```
Confusion Matrix:
 [[2 2 6]
 [5 5 0]
 [3 3 4]]
Akurasi: 0.36666666666666664
Presisi: [0.2 0.5 0.4]
Recall: [0.2 0.5 0.4]
F1 Score: [0.2 0.5 0.4]
```

In [ ]: