UAS PMPM GoogleNet

- Yosua Vito
- 220711893
- Keras
- Cabai (Rawit, Keriting, Hijau)
- MobileNet

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

data_dir = r"C:\Pembelajaran Mesin dan Pembelajaran Mendalam\UAS\
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.

['Hijau', 'Keriting', 'Rawit']
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')
```

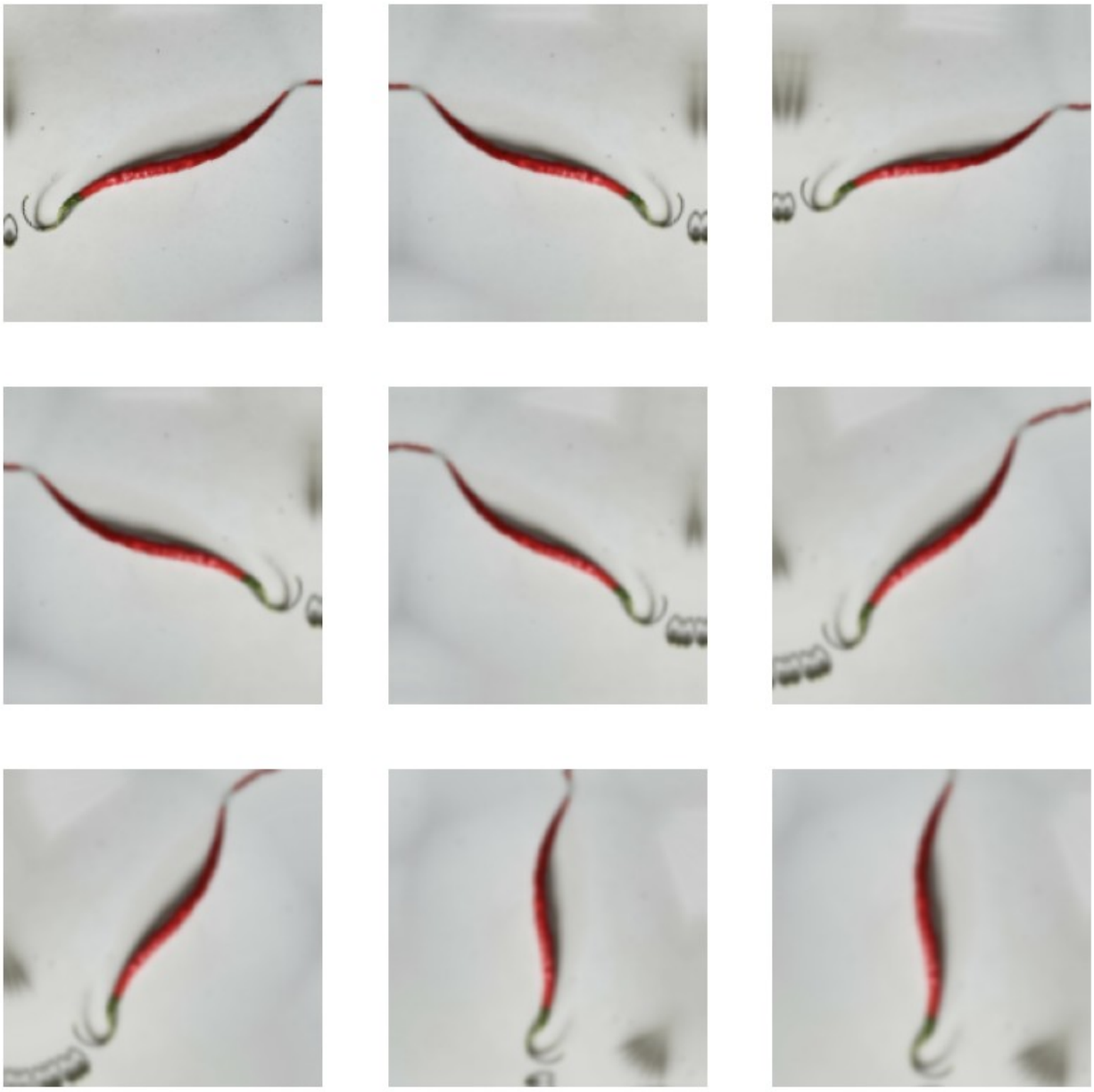

```
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

```
(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)
])

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



```
import tensorflow as tf
import keras

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

from keras._tf_keras.keras.models import load_model

#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
```

```
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,         │         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][0]… │
│                      │ 128)                │             │
│
├──────────────────────┼─────────────────────┼─────────────┼──────────────────
──────┤
│ conv2d_30 (Conv2D)   │ (None, 11, 11,      │      12,312 │
concatenate_3[0][0]… │
│                      │ 24)                 │             │
│
├──────────────────────┼─────────────────────┼─────────────┼──────────────────
──────┤
│ max_pooling2d_7      │ (None, 11, 11,      │           0 │
concatenate_3[0][0]… │
│ (MaxPooling2D)       │ 512)                │             │
│
├──────────────────────┼─────────────────────┼─────────────┼──────────────────
──────┤
│ conv2d_27 (Conv2D)   │ (None, 11, 11,      │      65,664 │
concatenate_3[0][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][0… |
├─────────────────────┼─────────────────────┼─────────────┼──────────────
──────┤
| conv2d_54 (Conv2D)  | (None, 6, 6, 48)  |     39,984 |
concatenate_7[0][0… |
├─────────────────────┼─────────────────────┼─────────────┼──────────────
──────┤
| max_pooling2d_12    | (None, 6, 6, 832) |          0 |
concatenate_7[0][0… |
| (MaxPooling2D)      |                     |            |
|
├─────────────────────┼─────────────────────┼─────────────┼──────────────
──────┤
| conv2d_51 (Conv2D)  | (None, 6, 6, 384) |    319,872 |
concatenate_7[0][0… |
├─────────────────────┼─────────────────────┼─────────────┼──────────────
──────┤
| conv2d_53 (Conv2D)  | (None, 6, 6, 384) |    663,936 | conv2d_52[0]
[0]   |
├─────────────────────┼─────────────────────┼─────────────┼──────────────
──────┤
```

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

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

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

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

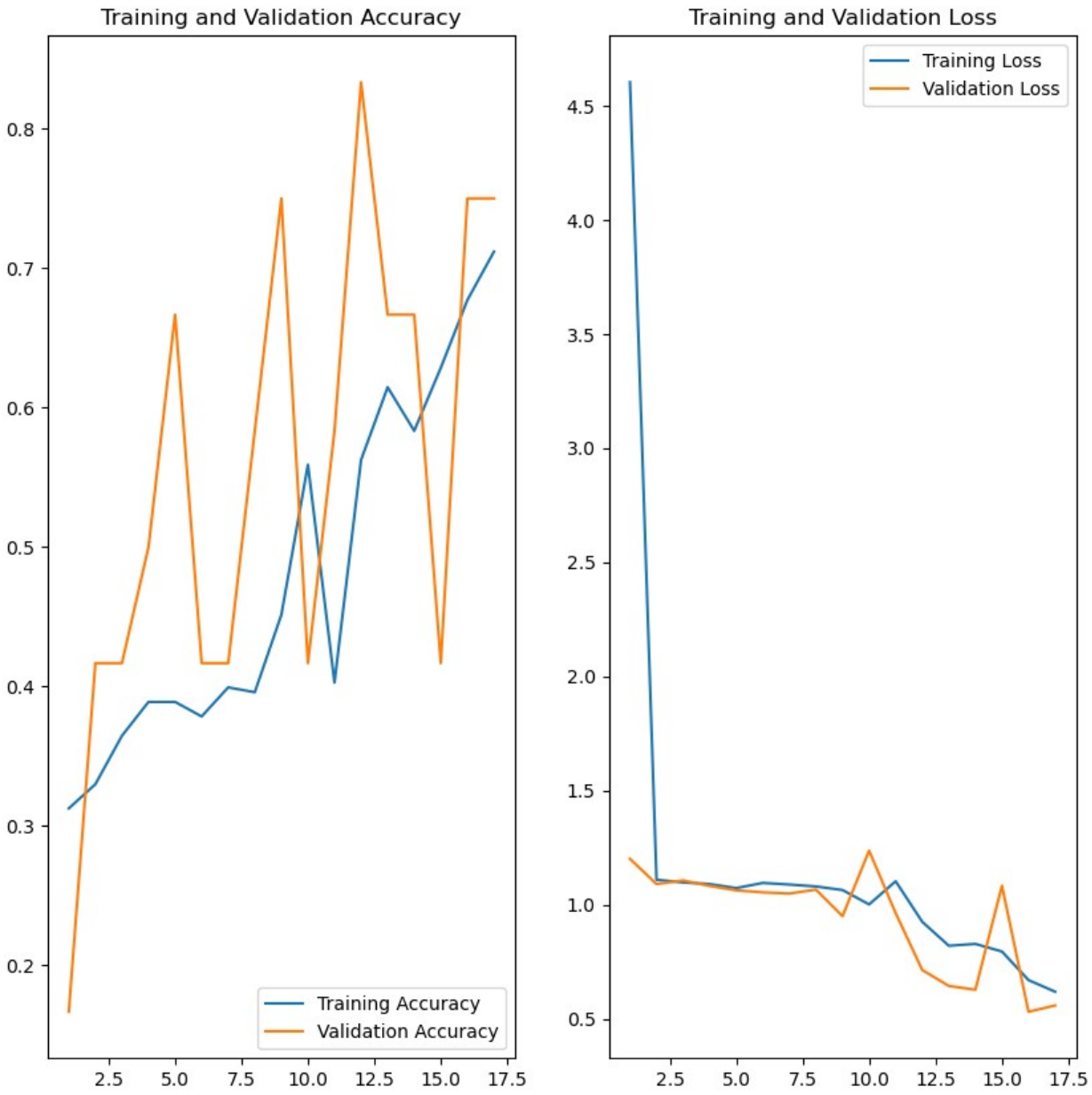
```
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

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
9/9 ━━━━━━━━━━━━━━━━━━━━ 17s 448ms/step - accuracy: 0.3010 - loss:
7.1376 - val_accuracy: 0.1667 - val_loss: 1.2006
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 311ms/step - accuracy: 0.3429 - loss:
1.1122 - val_accuracy: 0.4167 - val_loss: 1.0901
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 318ms/step - accuracy: 0.3261 - loss:
1.0981 - val_accuracy: 0.4167 - val_loss: 1.1060
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 318ms/step - accuracy: 0.3833 - loss:
1.0955 - val_accuracy: 0.5000 - val_loss: 1.0817
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 324ms/step - accuracy: 0.4019 - loss:
1.0733 - val_accuracy: 0.6667 - val_loss: 1.0625
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 343ms/step - accuracy: 0.3582 - loss:
1.0941 - val_accuracy: 0.4167 - val_loss: 1.0536
Epoch 7/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 341ms/step - accuracy: 0.3876 - loss:
1.0892 - val_accuracy: 0.4167 - val_loss: 1.0485
Epoch 8/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 3s 345ms/step - accuracy: 0.3950 - loss:
1.0828 - val_accuracy: 0.5833 - val_loss: 1.0657
Epoch 9/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 450ms/step - accuracy: 0.4322 - loss:
1.0707 - val_accuracy: 0.7500 - val_loss: 0.9494
Epoch 10/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 457ms/step - accuracy: 0.5452 - loss:
1.0371 - val_accuracy: 0.4167 - val_loss: 1.2364
Epoch 11/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 457ms/step - accuracy: 0.3856 - loss:
1.1730 - val_accuracy: 0.5833 - val_loss: 0.9627
Epoch 12/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 453ms/step - accuracy: 0.5800 - loss:
0.9144 - val_accuracy: 0.8333 - val_loss: 0.7136
Epoch 13/30
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 466ms/step - accuracy: 0.5946 - loss:
0.8362 - val_accuracy: 0.6667 - val_loss: 0.6437
Epoch 14/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 452ms/step - accuracy: 0.6180 - loss:
0.8131 - val_accuracy: 0.6667 - val_loss: 0.6268
Epoch 15/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 452ms/step - accuracy: 0.6496 - loss:
0.7223 - val_accuracy: 0.4167 - val_loss: 1.0827
Epoch 16/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 448ms/step - accuracy: 0.6867 - loss:
0.6895 - val_accuracy: 0.7500 - val_loss: 0.5305
Epoch 17/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 463ms/step - accuracy: 0.7507 - loss:
0.5507 - val_accuracy: 0.7500 - val_loss: 0.5579
```

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

```
# Load model Anda
model = load_model(r'./gugelnet.h5')  # Path ke model Anda
class_names = ['Cabe Hijau', 'Cabe Keriting', 'Cabe Rawit']

def classify_images(image_path):
    # Load dan preprocess gambar
    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)  #
Tambah dimensi batch

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

    return class_names[class_idx], confidence

def classify_all_images(folder_path, save_results=True):
    results = []  # Untuk menyimpan hasil prediksi
    for category in os.listdir(folder_path):  # Iterasi kategori
(Hijau, Keriting, Rawit)
        category_path = os.path.join(folder_path, category)
        if os.path.isdir(category_path):  # Pastikan folder
            for filename in os.listdir(category_path):  # Iterasi file
gambar
                if filename.endswith(('.jpg', '.jpeg', '.png')):  #
Filter hanya gambar
                    image_path = os.path.join(category_path, filename)
                    predicted_class, confidence =
classify_images(image_path)

                    # Simpan hasil
                    results.append({
                        'filename': filename,
                        'category': category,
                        'predicted_class': predicted_class,
                        'confidence': confidence
                    })

                    # Print hasil untuk setiap gambar
                    print(f"File: {filename} | Asli: {category} |
Prediksi: {predicted_class} | Confidence: {confidence:.2f}%")

    if save_results:
        # Simpan hasil prediksi ke file CSV
        import csv
```

```
        with open('classification_results.csv', 'w', newline='') as
csvfile:
            fieldnames = ['filename', 'category', 'predicted_class',
'confidence']
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
            writer.writeheader()
            writer.writerows(results)
        print("\nHasil klasifikasi disimpan di
'classification_results.csv'.")

    return results

# Jalankan fungsi untuk folder test_data
test_data_folder = r'./test_data'
results = classify_all_images(test_data_folder)
```

```
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 479ms/step
File: Hijau (103).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 56.15%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
File: Hijau (106).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 56.47%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
File: Hijau (107).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 54.87%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
File: Hijau (108).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.57%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
File: Hijau (109).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.60%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
File: Hijau (110).jpg | Asli: Hijau | Prediksi: Cabe Keriting |
Confidence: 43.59%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
File: Hijau (111).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.60%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
File: Hijau (112).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 50.44%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
File: Hijau (113).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.57%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
File: Hijau (114).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.55%
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
File: Hijau (115).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.28%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
File: Hijau (116).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 44.37%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Hijau (118).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.14%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
File: Hijau (119).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.45%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Hijau (120).jpg | Asli: Hijau | Prediksi: Cabe Keriting |
Confidence: 38.84%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
File: Hijau (121).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 47.18%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
File: Hijau (124).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.02%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
File: Hijau (125).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 56.70%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step
File: Hijau (126).jpg | Asli: Hijau | Prediksi: Cabe Hijau |
Confidence: 57.07%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Keriting (101).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.76%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
File: Keriting (102).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.28%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
File: Keriting (103).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 38.75%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
File: Keriting (104).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.71%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
File: Keriting (105).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 40.74%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
File: Keriting (106).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.66%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
File: Keriting (107).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.15%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
```

```
File: Keriting (108).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.36%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Keriting (109).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 40.43%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
File: Keriting (110).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.96%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
File: Keriting (111).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.09%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
File: Keriting (112).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.98%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
File: Keriting (113).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 40.28%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
File: Keriting (114).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.86%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
File: Keriting (115).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 38.50%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
File: Keriting (116).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.86%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Keriting (117).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.20%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
File: Keriting (118).jpg | Asli: Keriting | Prediksi: Cabe Rawit |
Confidence: 39.43%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Keriting (120).jpg | Asli: Keriting | Prediksi: Cabe Hijau |
Confidence: 43.61%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
File: Rawit (131).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 39.79%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Rawit (132).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 40.20%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
File: Rawit (133).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 39.82%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
File: Rawit (134).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 40.29%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
File: Rawit (135).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
```

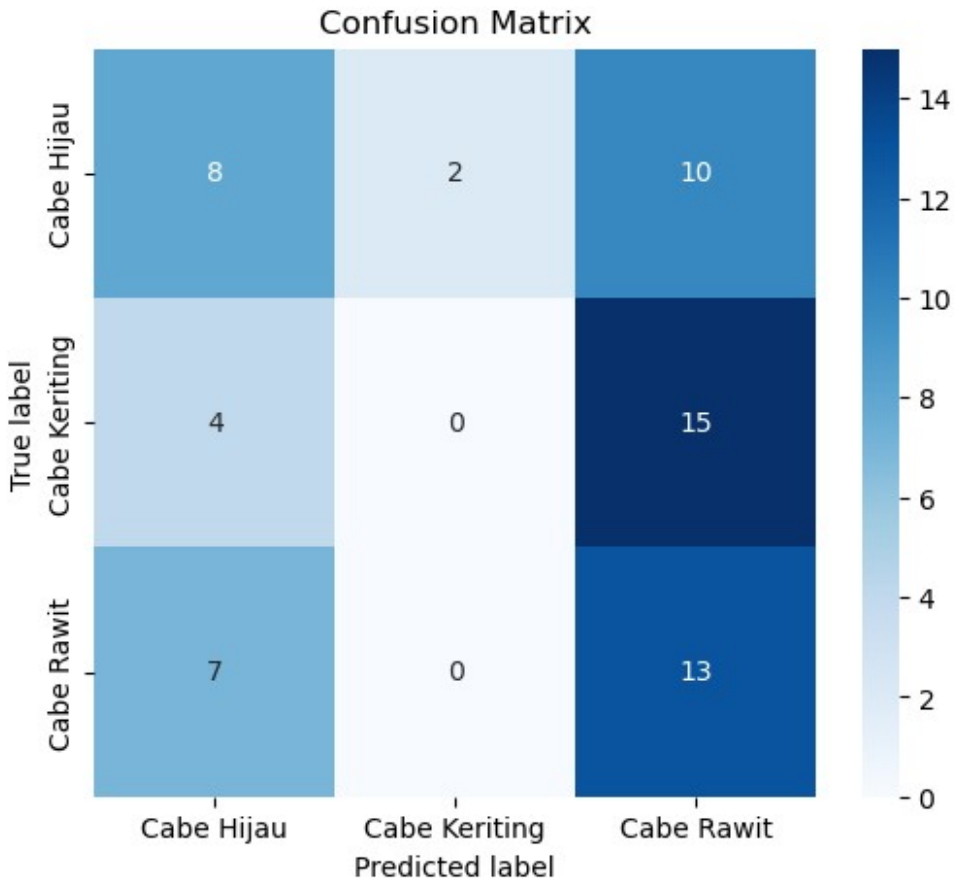
```
Confidence: 39.47%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
File: Rawit (136).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 40.21%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
File: Rawit (137).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 39.04%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
File: Rawit (138).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 39.02%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
File: Rawit (139).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 39.19%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
File: Rawit (140).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 38.81%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
File: Rawit (141).jpg | Asli: Rawit | Prediksi: Cabe Rawit |
Confidence: 39.50%

Hasil klasifikasi disimpan di 'classification_results.csv'.
```

```python
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,
num_classes=3)

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=["Cabe Hijau", "Cabe Keriting", "Cabe Rawit"],
yticklabels=["Cabe Hijau", "Cabe Keriting", "Cabe Rawit"])
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 59 files belonging to 3 classes.
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 907ms/step
```




```
Confusion Matrix:
 [[ 8  2 10]
 [ 4  0 15]
 [ 7  0 13]]
Akurasi: 0.3559322033898305
Presisi: [0.42105263 0.         0.34210526]
Recall: [0.4  0.   0.65]
F1 Score: [0.41025641        nan 0.44827586]
```

UAS PMPM Alexnet

- Ri'an
- 220711842
- Keras
- Cabai (Rawit, Keriting, Hijau)
- Alexnet

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

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

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

count = 0 #digunakan untuk menghitung jumlah gambar
dirs = os.listdir(r'D:\equalized\train_data')
for dir in dirs:
    files = list(os.listdir(r'D:\equalized\train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Hijau Folder has 100 Images
Keriting Folder has 100 Images
Rawit Folder has 100 Images
Images Folder has 300 Images
```

```
# Parameter
base_dir = r'D:\equalized\train_data' #direktori folder dataset
img_size = 224 #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
```

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

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

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

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

```
Class Names: ['Hijau', 'Keriting', 'Rawit']
```

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

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

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

```
###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)

import matplotlib.pyplot as plt

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

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

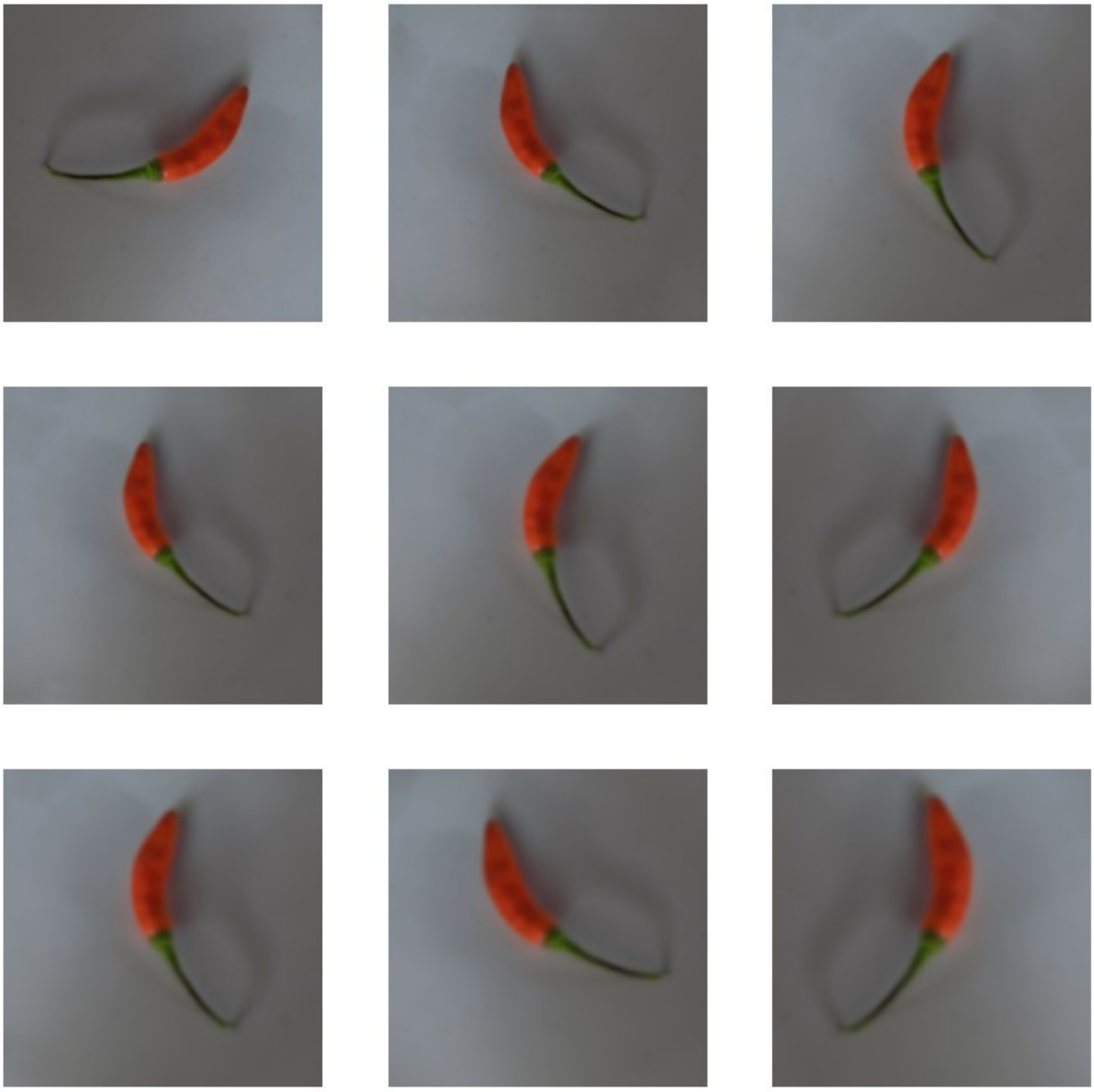

```
import numpy as np

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

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

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

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

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

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

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

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

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



```
from tensorflow.keras import layers, Sequential

def create_alexnet(input_shape, num_classes):
    model = Sequential([
        layers.Conv2D(96, kernel_size=(11, 11), strides=4,
activation='relu', input_shape=input_shape),
        layers.MaxPooling2D(pool_size=(3, 3), strides=2),
        layers.Conv2D(256, kernel_size=(5, 5), activation='relu',
padding='same'),
        layers.MaxPooling2D(pool_size=(3, 3), strides=2),
        layers.Conv2D(384, kernel_size=(3, 3), activation='relu',
padding='same'),
```

```
        layers.Conv2D(384, kernel_size=(3, 3), activation='relu',
padding='same'),
        layers.Conv2D(256, kernel_size=(3, 3), activation='relu',
padding='same'),
        layers.MaxPooling2D(pool_size=(3, 3), strides=2),
        layers.Flatten(),
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(4096, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model

# Contoh penggunaan
img_size = 224  # Misalnya
input_shape = (img_size, img_size, 3)
num_classes = len(class_names)  # Jumlah kelas yang Anda miliki

alexnet_model = create_alexnet(input_shape, num_classes)
```

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

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

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

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

```
Model: "sequential_5"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━
┩
```

```
━━━━━━┫
┃ conv2d_10 (Conv2D)                  ┃ (None, 54, 54, 96)        ┃
34,944 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ max_pooling2d_6 (MaxPooling2D)      ┃ (None, 26, 26, 96)        ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ conv2d_11 (Conv2D)                  ┃ (None, 26, 26, 256)       ┃
614,656 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ max_pooling2d_7 (MaxPooling2D)      ┃ (None, 12, 12, 256)       ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ conv2d_12 (Conv2D)                  ┃ (None, 12, 12, 384)       ┃
885,120 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ conv2d_13 (Conv2D)                  ┃ (None, 12, 12, 384)       ┃
1,327,488 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ conv2d_14 (Conv2D)                  ┃ (None, 12, 12, 256)       ┃
884,992 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ max_pooling2d_8 (MaxPooling2D)      ┃ (None, 5, 5, 256)         ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ flatten_2 (Flatten)                 ┃ (None, 6400)              ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ dense_6 (Dense)                     ┃ (None, 4096)              ┃
26,218,496 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ dropout_4 (Dropout)                 ┃ (None, 4096)              ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
┃ dense_7 (Dense)                     ┃ (None, 4096)              ┃
16,781,312 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━
━━━━━━┫
```

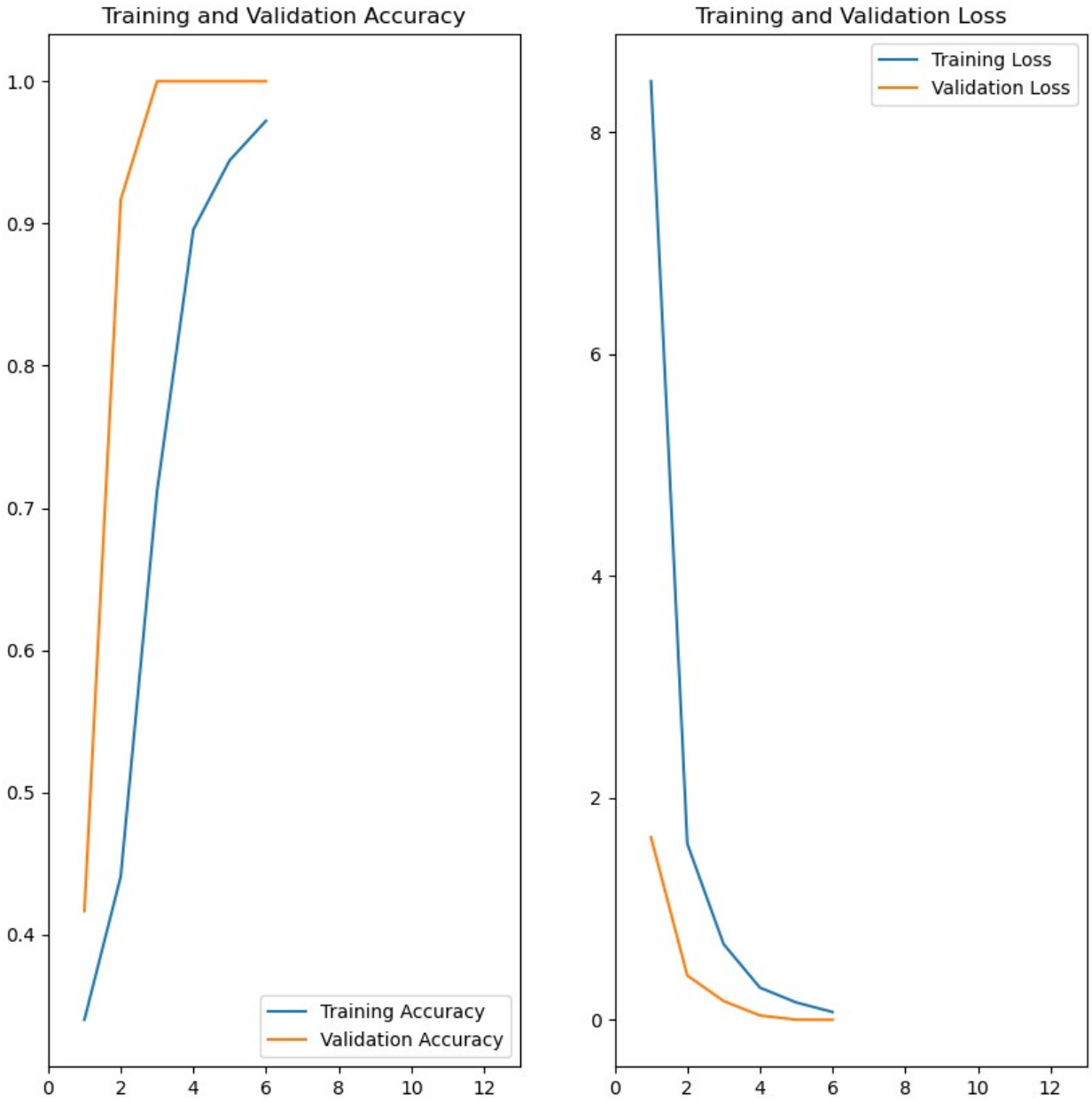
```
| dropout_5 (Dropout)                 | (None, 4096)              |
0 |
├─────────────────────────────────────┼───────────────────────────┼──────────
──────┤
| dense_8 (Dense)                     | (None, 3)                 |
12,291 |
└─────────────────────────────────────┴───────────────────────────┴──────────
──────┘

 Total params: 46,759,299 (178.37 MB)

 Trainable params: 46,759,299 (178.37 MB)

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

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

#Ada fungsi early stopping disini, jangan keskip tuan :D
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=3,
                               mode='max')
#melatih model menggunakan data latih dan validasi dengan early stopping
history= alexnet_model.fit(train_ds, #data pelatihan yang telah disiapkan
                   epochs=30, # jumlah maksimal epoch
                   validation_data=val_ds,  #data validasi untuk mengevaluasi model pada setiap epoch
                   callbacks=[early_stopping]) #menambahkan early stopping ke dalam callback untuk pelatihan
```

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 826ms/step - accuracy: 0.3171 - loss:
12.6633 - val_accuracy: 0.4167 - val_loss: 1.6458
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 8s 852ms/step - accuracy: 0.4081 - loss:
1.9028 - val_accuracy: 0.9167 - val_loss: 0.3998
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 8s 896ms/step - accuracy: 0.6249 - loss:
0.8143 - val_accuracy: 1.0000 - val_loss: 0.1700
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 8s 885ms/step - accuracy: 0.8572 - loss:
0.3739 - val_accuracy: 1.0000 - val_loss: 0.0398
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 8s 858ms/step - accuracy: 0.9284 - loss:
0.2062 - val_accuracy: 1.0000 - val_loss: 0.0021
Epoch 6/30
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 8s 844ms/step - accuracy: 0.9763 - loss:
0.0650 - val_accuracy: 1.0000 - val_loss: 0.0014
```

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

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

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

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




```
#menyimpan model yang telah dilatih
alexnet_model.save('model_alexnet.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
```

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

#memuat model yang sudah dilatih
alexnet_model = load_model(r'D:\equalized\model_alexnet.h5')  # Ganti dengan path model Anda
class_names = ['Hijau', 'Keriting', 'Rawit'] #kelas yang ada pada model

#fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        #memuat dan mempersiapkan gambar untuk prediksi
        input_image = tf.keras.utils.load_img(image_path, target_size=(224, 224)) #membuat gambar dari path dan mnegubah ukurannya menjadi 180x180 pixel
        input_image_array = tf.keras.utils.img_to_array(input_image) #mengubah gambar jadi array numpy agar bisa di proses model
        input_image_exp_dim = tf.expand_dims(input_image_array, 0) #menambahkan dimensi batch agar sesuai dengan input model

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

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

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

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

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

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

```
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.
```

```
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
alexnet_model = load_model(r'D:\equalized\model_alexnet.h5')#gunakan path masing masing ya

#memuat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data', #direktori data uji
    labels='inferred', #label otomatis dari subfolder yang ada
    label_mode='categorical',  #menghasilkan label dalam bentuk one-hot encoding
    batch_size=32, #ukuran batch untuk pemrosesan
    image_size=(224, 224) #ukuran gambar yang akan diproses
)

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

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

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

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

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

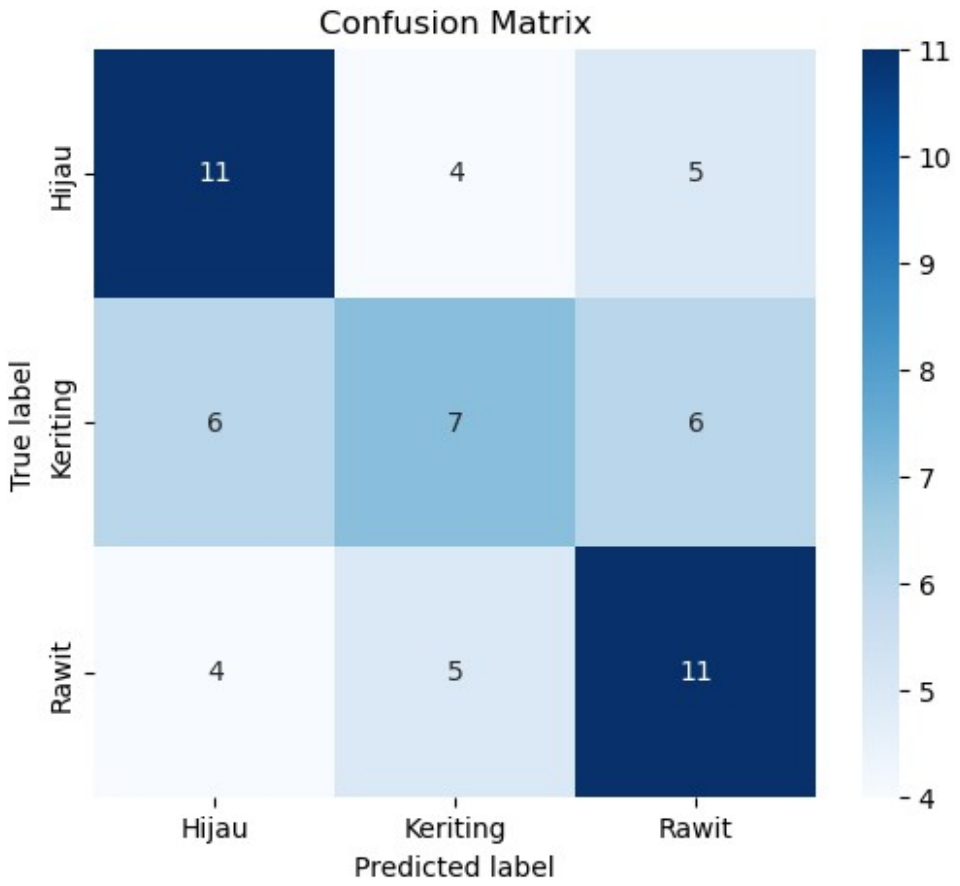
```
axis=1)

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

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

#fmt='d' untuk menampilkan bilangan bulat tanpa desimal
            xticklabels=["Hijau", "Keriting", "Rawit"],
yticklabels=["Hijau", "Keriting", "Rawit"])
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 built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

```
Found 59 files belonging to 3 classes.
WARNING:tensorflow:5 out of the last 5 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x000002A5BAC440E0> 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:5 out of the last 5 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x000002A5BAC440E0> 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.

1/2 ━━━━━━━━━━━━━━━━━━━━ 0s 450ms/stepWARNING:tensorflow:6 out of the
last 6 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x000002A5BAC440E0> 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_dist
ributed at 0x000002A5BAC440E0> 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.

2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 323ms/step
```




```
Confusion Matrix:
 [[11  4  5]
 [ 6  7  6]
 [ 4  5 11]]
Akurasi: 0.4915254237288136
Presisi: [0.52380952 0.4375     0.5       ]
Recall: [0.55       0.36842105 0.55      ]
F1 Score: [0.53658537 0.4        0.52380952]
```

UAS PMPM VGG-16

- Vinciant Andra Kaisarea
- 220711902
- Keras
- Cabai (Rawit, Keriting, Hijau)
- VGG-16

In [22]:
```
#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
```

In [23]:
```
count = 0 #digunakan untuk menghitung jumlah gambar
dirs = os.listdir(r"D:\UAJY Kuliah Andra 220711902\Matkul\Semester 5\Pembelajaran M
for dir in dirs:
    files = list(os.listdir(r"D:\UAJY Kuliah Andra 220711902\Matkul\Semester 5\Pemb
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Hijau Folder has 100 Images
Keriting Folder has 100 Images
Rawit Folder has 100 Images
Images Folder has 300 Images
```

In [24]:
```
# Parameter
base_dir = r"D:\UAJY Kuliah Andra 220711902\Matkul\Semester 5\Pembelajaran Mesin da
img_size = 224 #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 [25]:
```
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 224x224
    batch_size=batch, #jumlah gambar yang akan dikelompokkan
)
```

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

In [26]:
```
#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: ['Hijau', 'Keriting', 'Rawit']
```

In [27]:
```
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: 10
Train Images: 9
Validation Images: 1
```

In [28]:
```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count)
```

In [29]:
```
import matplotlib.pyplot as plt

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


for images, labels in train_ds.take(1): #mengambil 1 batch pertama dari train_ds
    for i in range(9):
        plt.subplot(3,3, i+1) #menyiapkan subplot dengan grid 3x3 dan menempatkan g
        plt.imshow(images[i].numpy().astype('uint8')) #menampilkan gambar dan mengo
        plt.title(class_names[labels[i]]) #menampilkan judul gambar sesuai dengan n
        plt.axis('off') #menonaktifkan sumbu pada gambar agar tidak terlihat
```

In [30]:
```
import numpy as np

# Tampilkan gambar dengan shape
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)  # Output
```

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

In [31]:
```
AUTOTUNE = tf.data.AUTOTUNE
```

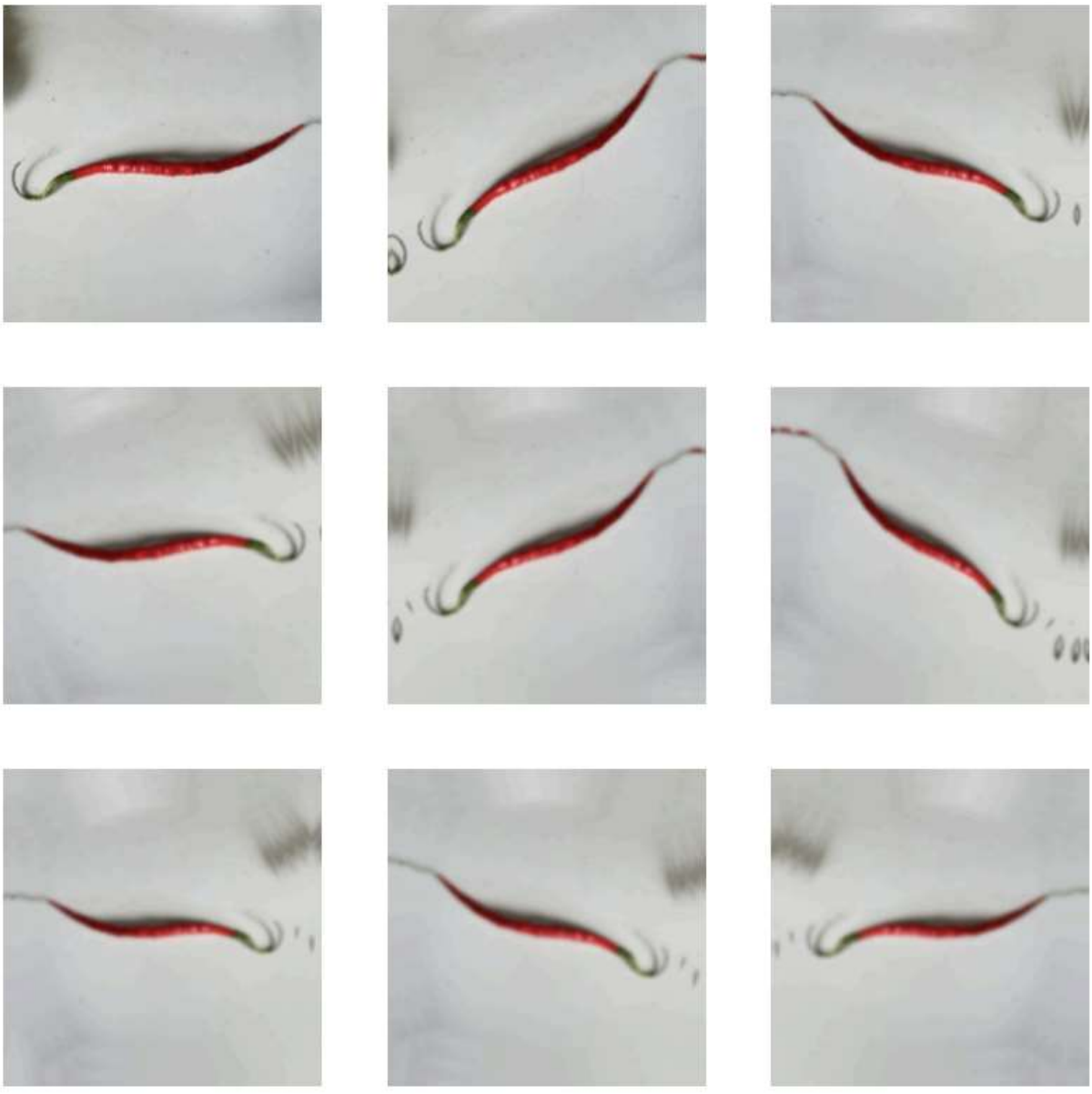
In [32]:
```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

In [33]:
```
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

In [34]:
```
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
])
```

In [35]:
```
#sama seperti sebelumnya, code ini digunakan untuk menampilkan gambar dari data_aug
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 [36]:
```
from tensorflow.keras import layers, Sequential

def create_vgg16(input_shape, num_classes):
    model = Sequential([
```

```
        # Input Layer
        layers.Input(shape=input_shape),

        # Convolutional Block 1
        layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
        layers.Conv2D(64, (3, 3), padding='same', activation='relu'),
        layers.MaxPooling2D((2, 2), strides=(2, 2)),

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

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

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

        # Convolutional Block 5
        layers.Conv2D(512, (3, 3), padding='same', activation='relu'),
        layers.Conv2D(512, (3, 3), padding='same', activation='relu'),
        layers.Conv2D(512, (3, 3), padding='same', activation='relu'),
        layers.MaxPooling2D((2, 2), strides=(2, 2)),

        # Fully Connected Layers
        layers.Flatten(),
        layers.Dense(4096, activation='relu'),
        layers.Dense(4096, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model

# Contoh penggunaan
img_size = 224
input_shape = (img_size, img_size, 3)
num_classes = len(class_names)  # Jumlah kelas

vgg16_model = create_vgg16(input_shape, num_classes)
```

In [37]:

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

#mengkompilasi model dengan optimizer, loss function, dan metrics
vgg16_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 [38]:
```
#menampilkan ringkasan dari model
vgg16_model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36,928
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73,856
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147,584
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295,168
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590,080
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590,080
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_7 (Conv2D)	(None, 28, 28, 512)	1,180,160
conv2d_8 (Conv2D)	(None, 28, 28, 512)	2,359,808
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2,359,808
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_10 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4096)	102,764,544
dense_1 (Dense)	(None, 4096)	16,781,312
dense_2 (Dense)	(None, 3)	12,291

Total params: 134,272,835 (512.21 MB)

Trainable params: 134,272,835 (512.21 MB)

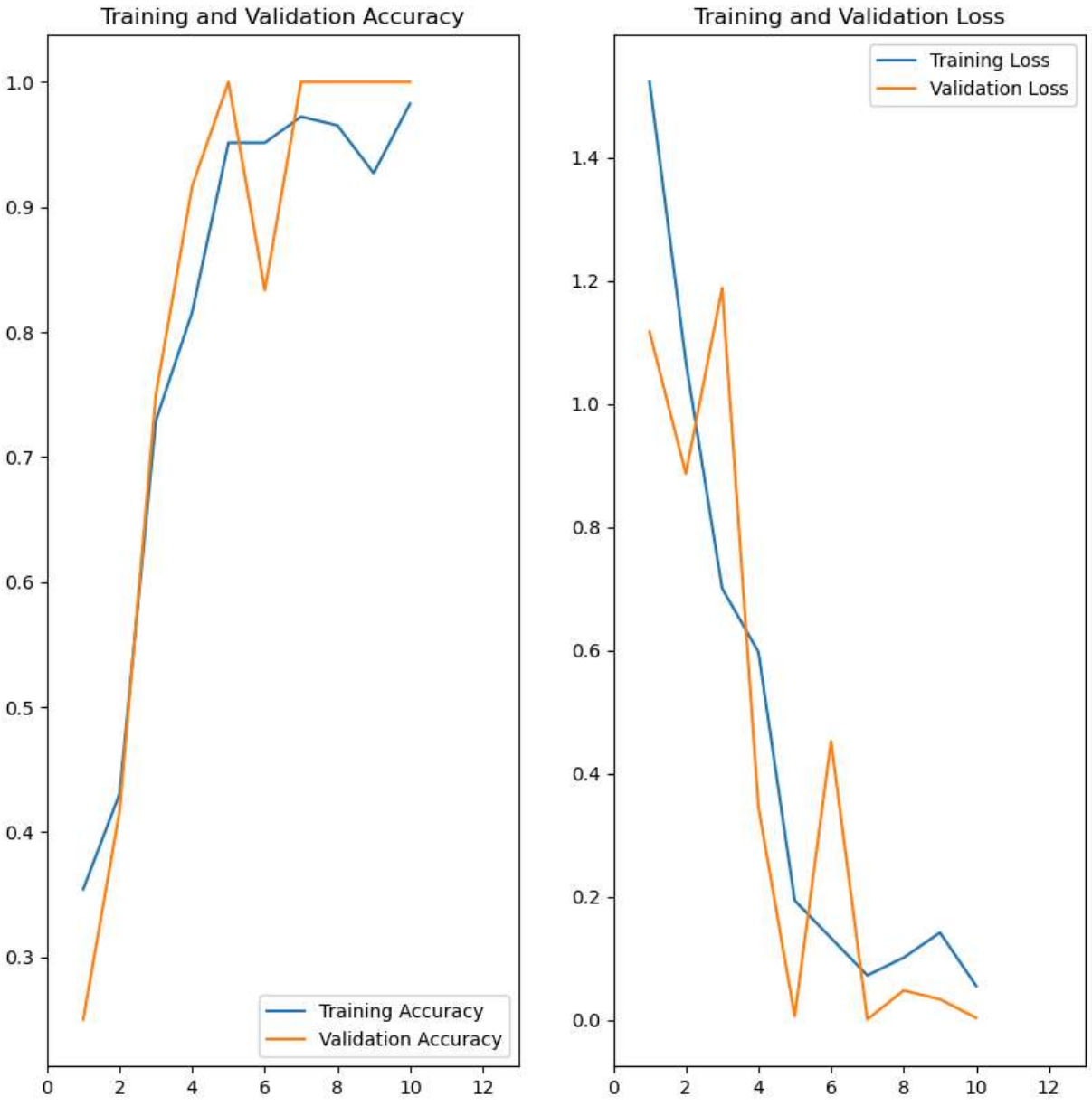
Non-trainable params: 0 (0.00 B)

In [39]:
```
#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=5,
                               mode='max')
#melatih model menggunakan data latih dan validasi dengan early stopping
history= vgg16_model.fit(train_ds, #data pelatihan yang telah disiapkan
                  epochs=40, # jumlah maksimal epoch
                  validation_data=val_ds,  #data validasi untuk mengevaluasi model
                  callbacks=[early_stopping]) #menambahkan early stopping ke dalam
```

```
Epoch 1/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 40s 4s/step - accuracy: 0.3575 - loss: 1.8235 - val_accurac
y: 0.2500 - val_loss: 1.1173
Epoch 2/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.4128 - loss: 1.0755 - val_accurac
y: 0.4167 - val_loss: 0.8868
Epoch 3/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.6201 - loss: 0.8589 - val_accurac
y: 0.7500 - val_loss: 1.1883
Epoch 4/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 34s 4s/step - accuracy: 0.7585 - loss: 0.7605 - val_accurac
y: 0.9167 - val_loss: 0.3455
Epoch 5/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.9521 - loss: 0.2479 - val_accurac
y: 1.0000 - val_loss: 0.0059
Epoch 6/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.9420 - loss: 0.1407 - val_accurac
y: 0.8333 - val_loss: 0.4521
Epoch 7/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.9599 - loss: 0.0962 - val_accurac
y: 1.0000 - val_loss: 5.3107e-04
Epoch 8/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.9756 - loss: 0.0752 - val_accurac
y: 1.0000 - val_loss: 0.0475
Epoch 9/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.9204 - loss: 0.1588 - val_accurac
y: 1.0000 - val_loss: 0.0332
Epoch 10/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 35s 4s/step - accuracy: 0.9814 - loss: 0.0620 - val_accurac
y: 1.0000 - val_loss: 0.0029
```

In [40]:
```
#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 [41]:
```
#menyimpan model yang telah dilatih
vgg16_model.save('BestModel_VGG16_Keras.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')`.
```

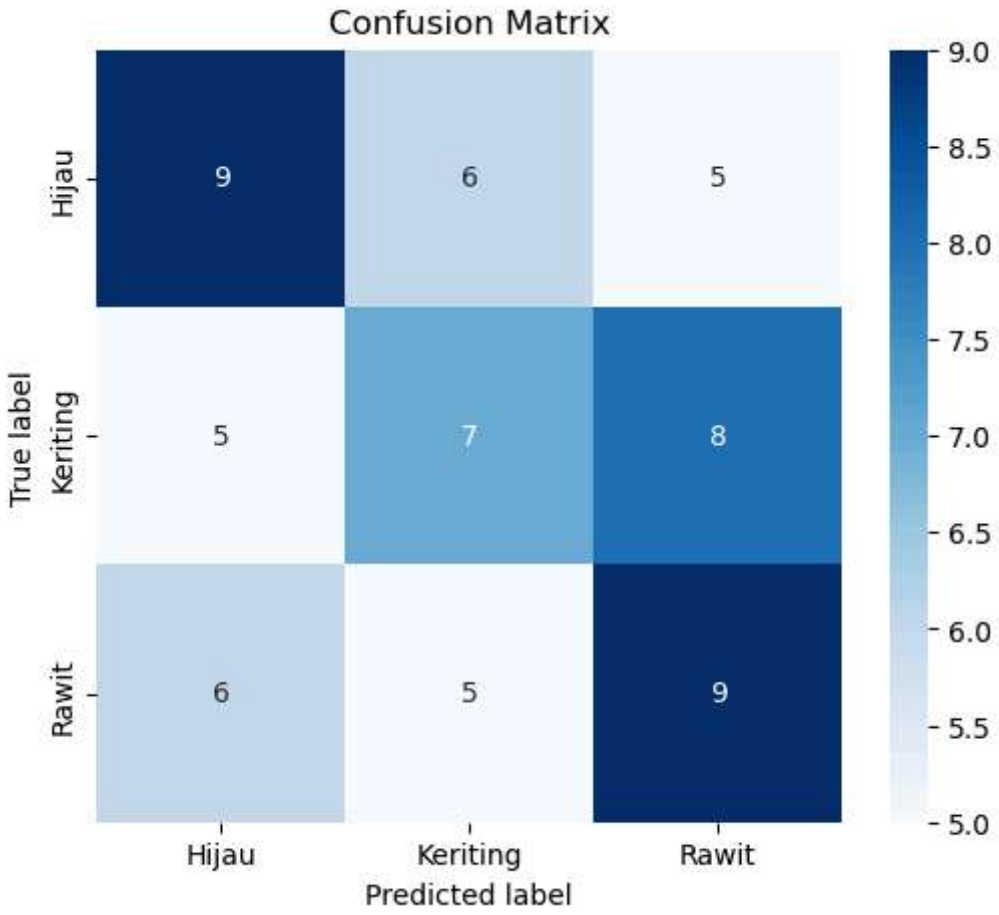
In [43]:
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

#memuat model yang sudah dilatih
vgg16_model = load_model(r"D:\UAJY Kuliah Andra 220711902\Matkul\Semester 5\Pembela
class_names = ['Hijau', 'Keriting', 'Rawit'] #kelas yang ada pada model

#fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        #memuat dan mempersiapkan gambar untuk prediksi
        input_image = tf.keras.utils.load_img(image_path, target_size=(224, 224)) #
        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


        #melakukan prediksi
        predictions = vgg16_model.predict(input_image_exp_dim) #melakukan prediksi
        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}"

result = classify_images(r"D:\UAJY Kuliah Andra 220711902\Matkul\Semester 5\Pembela
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.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step
Prediksi: Keriting
Confidence: 57.50%
Prediksi: Keriting dengan confidence 57.50%. Gambar asli disimpan di keritingTest.jp
g.
```

In [94]:
```
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
vgg16_model = load_model(r"D:\UAJY Kuliah Andra 220711902\Matkul\Semester 5\Pembela
```

```
#memuat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data', #direktori data uji
    labels='inferred', #label otomatis dari subfolder yang ada
    label_mode='categorical',  #menghasilkan label dalam bentuk one-hot encoding
    batch_size=32, #ukuran batch untuk pemrosesan
    image_size=(224, 224) #ukuran gambar yang akan diproses
)

#prediksi 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
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=["Hijau", "Keriting", "Rawit"], yticklabels=["Hijau", "Keri
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 60 files belonging to 3 classes.
2/2 ━━━━━━━━━━━━━━━━━━━━ 2s 1s/step
```




```
Confusion Matrix:
 [[9 6 5]
 [5 7 8]
 [6 5 9]]
Akurasi: 0.4166666666666667
Presisi: [0.45       0.38888889 0.40909091]
Recall: [0.45 0.35 0.45]
F1 Score: [0.45       0.36842105 0.42857143]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

UAS PMPM MobileNet

- Aldo
- 220711837
- Keras
- Cabai (Rawit, Keriting, Hijau)
- MobileNet

In [202]:
```
import os
import numpy as np

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import load_img, ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
```

In [203]:
```
count = 0
dirs = os.listdir(r'C:\Users\Aldo\Downloads\equalized\train_data')
for dir in dirs:
    files = list(os.listdir(r'C:\Users\Aldo\Downloads\equalized\train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Hijau Folder has 100 Images
Keriting Folder has 100 Images
Rawit Folder has 100 Images
Images Folder has 300 Images
```

In [204]:
```
base_dir = r'C:\Users\Aldo\Downloads\equalized\train_data'
img_size = 224
batch = 32
validation_split = 0.1
```

In [205]:
```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
)
```

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

In [206]:
```
class_names = dataset.class_names
print("Class Names:", class_names)
```

```
Class Names: ['Hijau', 'Keriting', 'Rawit']
```

In [207]:
```
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: 10
Train Images: 9
Validation Images: 1
```

In [208]:
```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count)
```

In [209]:
```
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 [210]:
```
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 [211]:
```
AUTOTUNE = tf.data.AUTOTUNE
```

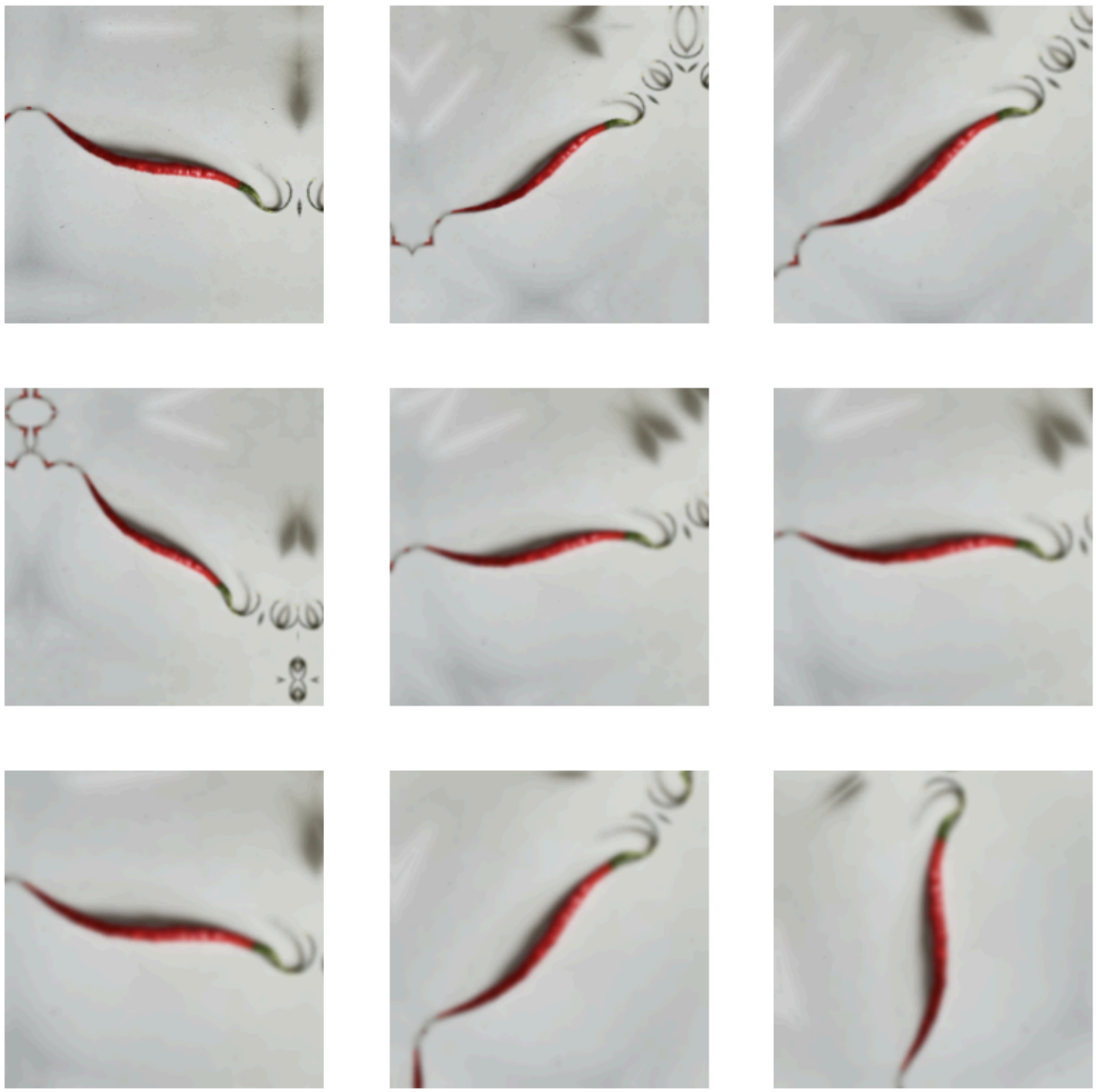
In [212]:
```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

In [213]:
```
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

In [214]:
```
data_augmentation = Sequential([
    layers.RandomFlip("diagonal", input_shape = (img_size,img_size,3)),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2)
])
```

```
C:\Users\Aldo\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\prepr
ocessing\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)` o
bject as the first layer in the model instead.
  super().__init__(**kwargs)
```

In [215]:
```
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 [216]:
```
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model

base_model = MobileNet(include_top=False, input_shape=(img_size, img_size, 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,
    layers.GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(len(class_names),activation='softmax')

])
```

In [217]:
```
from tensorflow.keras.optimizers import Adam
```

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

In [218]:
```
model.summary()
```

Model: "sequential_19"

Layer (type)	Output Shape	Para
sequential_18 (Sequential)	(None, 224, 224, 3)	
rescaling_9 (Rescaling)	(None, 224, 224, 3)	
mobilenet_1.00_224 (Functional)	(None, 7, 7, 1024)	3,228,
global_average_pooling2d_9 (GlobalAveragePooling2D)	(None, 1024)	
dense_18 (Dense)	(None, 128)	131,
dropout_9 (Dropout)	(None, 128)	
dense_19 (Dense)	(None, 3)	

Total params: 3,360,451 (12.82 MB)

Trainable params: 3,069,443 (11.71 MB)

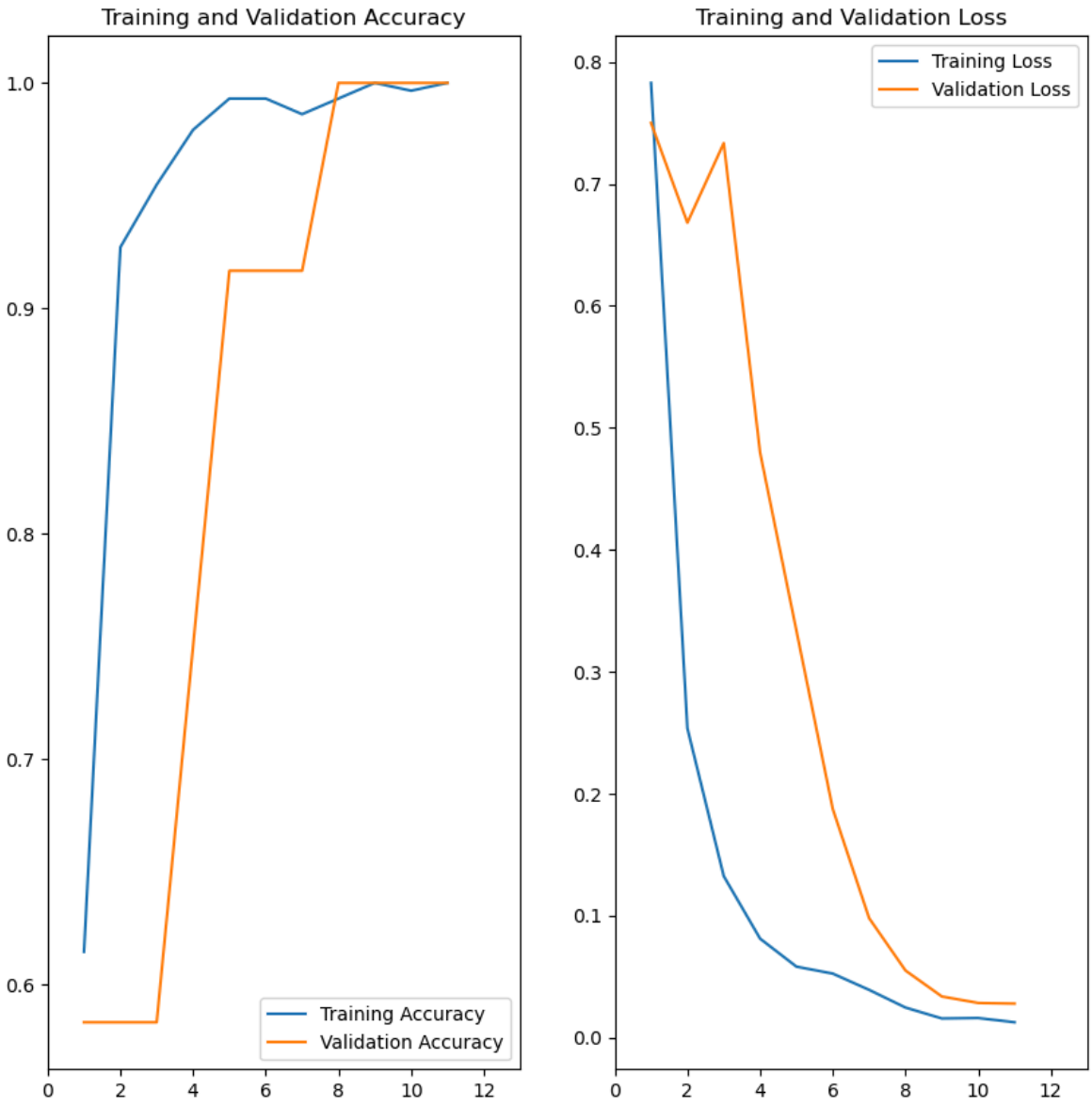
Non-trainable params: 291,008 (1.11 MB)

In [219]:
```
from tensorflow.keras.callbacks import EarlyStopping

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

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

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 56s 4s/step - accuracy: 0.4944 - loss: 0.9520 - val_accura
cy: 0.5833 - val_loss: 0.7501
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 0.9104 - loss: 0.2984 - val_accura
cy: 0.5833 - val_loss: 0.6683
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9677 - loss: 0.1243 - val_accura
cy: 0.5833 - val_loss: 0.7335
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9846 - loss: 0.0683 - val_accura
cy: 0.7500 - val_loss: 0.4797
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9913 - loss: 0.0662 - val_accura
cy: 0.9167 - val_loss: 0.3336
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9974 - loss: 0.0416 - val_accura
cy: 0.9167 - val_loss: 0.1874
Epoch 7/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9812 - loss: 0.0516 - val_accura
cy: 0.9167 - val_loss: 0.0979
Epoch 8/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9888 - loss: 0.0267 - val_accura
cy: 1.0000 - val_loss: 0.0550
Epoch 9/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 1.0000 - loss: 0.0160 - val_accura
cy: 1.0000 - val_loss: 0.0337
Epoch 10/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9973 - loss: 0.0138 - val_accura
cy: 1.0000 - val_loss: 0.0284
Epoch 11/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 1.0000 - loss: 0.0117 - val_accura
cy: 1.0000 - val_loss: 0.0279
```

In [220]:

```
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 [221]:
```
model.save('model_mobilenet.h5')
```

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

In [222]:
```
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\Aldo\Downloads\Tugas6_B_Aldo_11837\Tugas6_B_Aldo_118
class_names = ['Hijau', 'Keriting','Rawat']

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'test_data/Keriting/Keriting (101).jpg', save_path='keri
print(result)
```

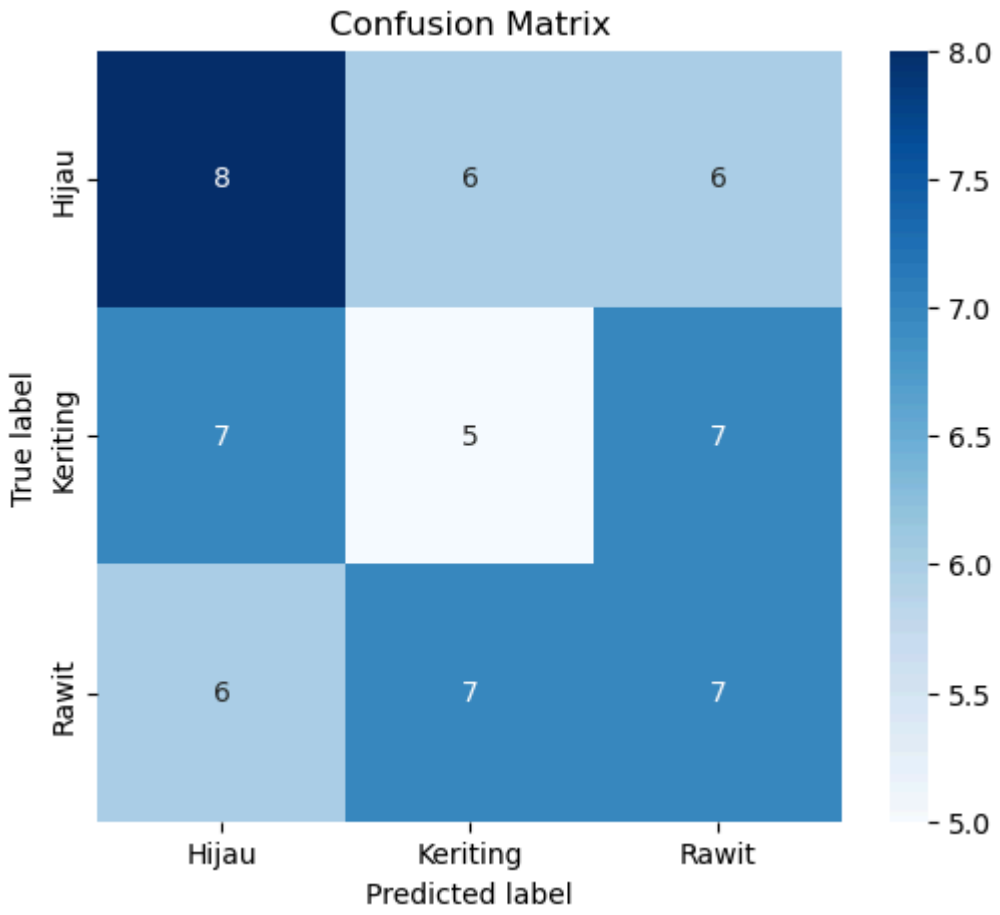
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be bui
lt. `model.compile_metrics` will be empty until you train or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Prediksi: Keriting
Confidence: 52.05%
Prediksi: Keriting dengan confidence 52.05%. Gambar asli disimpan di keriting.jpg.
```

In [223]:

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

mobileNet_model = load_model(r'C:\Users\Aldo\Downloads\Tugas6_B_Aldo_11837\Tugas6_

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(224, 224)
)

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

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

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

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

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

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

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Hijau", "Keriting", "Rawit"], yticklabels=["Hijau", "Ker
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 bui
lt. `model.compile_metrics` will be empty until you train or evaluate the model.
Found 59 files belonging to 3 classes.
2/2 ━━━━━━━━━━━━━━━━━━━━ 4s 2s/step
```




```
Confusion Matrix:
 [[8 6 6]
 [7 5 7]
 [6 7 7]]
Akurasi: 0.3389830508474576
Presisi: [0.38095238 0.27777778 0.35      ]
Recall: [0.4        0.26315789 0.35      ]
F1 Score: [0.3902439  0.27027027 0.35      ]
```

MainStreamlit_B_Keras.py

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

model = load_model(r'model_alexnet.h5')
class_names = ['Hijau', 'Keriting', 'Rawit']

def classify_image(image_path):
    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_scores= result.numpy()
        return class_names[class_idx], confidence_scores
    except Exception as e:
        return "Error", str(e)

def custom_progress_bar(confidence, color1, color2, color3):
    percentage1 = confidence[0] * 100
    percentage2 = confidence[1] * 100
    percentage3 = confidence[2] * 100
    progress_html = f"""
    <div style="border: 1px solid #ddd; border-radius: 5px;
overflow:hidden; width: 100%; font-size: 14px;">
```

```
        <div style="width: {percentage1:.2f}%; background: {color1}; color:
white; text-align: center; height: 24px; float: left;">

            {percentage1:.2f}%

        </div>

        <div style="width: {percentage2:.2f}%; background: {color2}; color:
white; text-align: center; height: 24px; float: left;">

            {percentage2:.2f}%

        </div>

        <div style="width: {percentage3:.2f}%; background: {color3}; color:
white; text-align: center; height: 24px; float: left;">

            {percentage3:.2f}%

        </div>

    </div>

    """

    st.sidebar.markdown(progress_html, unsafe_allow_html=True)


st.title("Prediksi Jenis Cabai (Hijau, Keriting merah, Rawit Merah)\n")

st.text("Kelompok Keras\n")

st.text("Aldo 220711837\nRian 220711842\nYosua Vito 220711893\nVinciant
Andra Kaisarea 220711902")


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


if st.sidebar.button("Prediksi"):

    if uploaded_files:

        st.sidebar.write("### Hasil Prediksi")

        for uploaded_file in uploaded_files:

            with open(uploaded_file.name, "wb") as f:

                f.write(uploaded_file.getbuffer())


            label, confidence = classify_image(uploaded_file.name)


            if label != "Error":

                primary_color = "#00FF00"
```

```
                secondary_color = "#FF4136"

                Third_color = "#FFA500"

                label_color = primary_color if label == "Hijau" else
secondary_color if label == "Keriting" else Third_color


                st.sidebar.write(f"**nama File:** {uploaded_file.name}")

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


                st.sidebar.write("**Confidence:**")

                for i, class_name in enumerate(class_names):

                    st.sidebar.write(f"- {class_name}: {confidence[i] *
100:.2f}%")


                custom_progress_bar(confidence, primary_color,
secondary_color, Third_color)


                st.sidebar.write("---")

            else:

                st.sidebar.error(f"Kesalahan saat memproses gambar
{uploaded_file.name}: {confidence}")

    else:

        st.sidebar.error("Silahkan unggah setidaknya satu gambar untuk
diprediksi.")


if uploaded_files:

    st.write("### Preview Gambar")

    for uploaded_file in uploaded_files:

        image = Image.open(uploaded_file)

        st.image(image, caption=f"{uploaded_file.name}",
use_column_width=True)
```