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

model =
load_model(r'C:\Users\ASUS\Desktop\Folder pengumpulan\dataset\BestModel_VGG
CNN_Tensorflow.h5')
class_names = ["Jahe", "Kunyit",
"Lengkuas"]

def predict_image(image):
    image = image.resize((224, 224))

image_array = np.array(image) / 255.0
    image_array = np.expand_dims(image_array, axis=0)

  predictions = model.predict(image_array)
    predicted_class = np.argmax(predictions)

confidence = np.max(predictions)
    return class_names[predicted_class],
confidence

st.set_page_config(page_title="Klasifikasi Rempah",
page_icon="?", layout="centered")
st.title("? Klasifikasi
Rempah")
st.write("Unggah gambar rempah seperti Jahe, Lengkuas, atau Kunyit, dan
model kami akan mengidentifikasinya.")

uploaded_file = st.file_uploader("Unggah
gambar", type=["jpg", "jpeg", "png"])

if uploaded_file is
not None:
    image = Image.open(uploaded_file)
    st.image(image, caption="Gambar yang
diunggah", use_column_width=True)

    if st.button("Prediksi Gambar"):

  with st.spinner("Sedang memproses gambar..."):
            label, confidence =
predict_image(image)

        st.success("Prediksi Selesai!")

st.write(f"**Label:** {label}")
        st.write(f"**Kepercayaan:** {confidence
* 100:.2f}%")

st.markdown("---")
```

```
import os
import numpy as np

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout, DepthwiseConv2D, BatchNormalization, ReLU,
GlobalAveragePooling2D, SeparableConv2D
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau, ModelCheckpoint

count = 0
dirs = os.listdir(r'C:\Users\ASUS\Documents\dataset\Mobile Net\train')
for i in dirs:
    files = list(os.listdir(r'C:\Users\ASUS\Documents\dataset\Mobile
Net\train' + '/' + i))
    count += len(files)
    print(i + ' punya ' + str(len(files)) + ' images')
print('Total gambar dalam folder: ', count)
```

```
jahe punya 100 images
kunyit punya 100 images
lengkuas punya 100 images
Total gambar dalam folder:  300
```

```
import os
import shutil
from sklearn.model_selection import train_test_split

base_dir = r"C:\Users\ASUS\Documents\dataset\Mobile Net\train"
validation_dir = r"C:\Users\ASUS\Documents\dataset\Mobile Net\
validation"
test_dir = r"C:\Users\ASUS\Documents\dataset\Mobile Net\test"

os.makedirs(validation_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)

categories = [name for name in os.listdir(base_dir) if
os.path.isdir(os.path.join(base_dir, name))]

validation_split = 0.1  # 10% untuk validasi
test_split = 0.1        # 10% untuk test
train_split = 1 - validation_split - test_split  # 80% untuk train

for category in categories:
    category_dir = os.path.join(base_dir, category)
    validation_category_dir = os.path.join(validation_dir, category)
    test_category_dir = os.path.join(test_dir, category)
```

```
    os.makedirs(validation_category_dir, exist_ok=True)
    os.makedirs(test_category_dir, exist_ok=True)

    all_files = os.listdir(category_dir)

    train_val_files, test_files = train_test_split(all_files,
test_size=test_split, random_state=42)

    train_files, val_files = train_test_split(train_val_files,
test_size=validation_split / (train_split + validation_split),
random_state=42)

    for file in val_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(validation_category_dir, file))

    for file in test_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(test_category_dir, file))

print("Dataset berhasil dibagi menjadi train, validation, dan test!")
```

```
Dataset berhasil dibagi menjadi train, validation, dan test!
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=45,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    zoom_range=0.3,
    brightness_range=[0.8, 1.2],
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode="nearest"
)

validation_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
```

```
    r'train',
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical',
    shuffle=True
)

validation_generator = validation_datagen.flow_from_directory(
    r'validation',
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical',
    shuffle=True
)

test_generator = test_datagen.flow_from_directory(
    r'test',
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical'
)
```

```
Found 240 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
```

```
sample_batch = next(train_generator)

plt.figure(figsize=(10, 10))
for i in range(min(9, len(sample_batch[0]))):
    image = sample_batch[0][i]
    label = sample_batch[1][i]
    plt.subplot(3, 3, i + 1)
    plt.imshow(image)
    plt.title(f"Label: {label}\nSize: {image.shape}")
    plt.axis('off')
plt.show()
```


```
def create_mobilenet_model(img_size, num_classes, depth_multiplier=1):
    model = Sequential()

    # Convolutional layers
    model.add(SeparableConv2D(int(32 * depth_multiplier), (3, 3),
activation='relu', padding='same', input_shape=(img_size, img_size,
3)))
    model.add(BatchNormalization())
    model.add(Conv2D(int(32 * depth_multiplier), (1, 1),
activation='relu', padding='same', kernel_regularizer='l2'))
    model.add(BatchNormalization())
```

```
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(DepthwiseConv2D((3, 3), activation='relu',
padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(int(64 * depth_multiplier), (1, 1),
activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(DepthwiseConv2D((3, 3), activation='relu',
padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(int(128 * depth_multiplier), (1, 1),
activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(DepthwiseConv2D((3, 3), activation='relu',
padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(int(256 * depth_multiplier), (1, 1),
activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(GlobalAveragePooling2D())
    model.add(Dense(int(512 * depth_multiplier), activation='relu',
kernel_initializer='he_normal', kernel_regularizer='l2'))
    model.add(Dropout(0.7))
    model.add(Dense(num_classes, activation='softmax'))

    return model

model = create_mobilenet_model(img_size=224, num_classes=3)
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\layers\convolutional\base_separable_conv.py:104: 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__(
```

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

model.summary()
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ separable_conv2d                     │ (None, 224, 224, 32)        │             155 │
│ (SeparableConv2D)                    │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization                  │ (None, 224, 224, 32)        │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d (Conv2D)                      │ (None, 224, 224, 32)        │           1,056 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_1                │ (None, 224, 224, 32)        │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 112, 112, 32)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ depthwise_conv2d                     │ (None, 112, 112, 32)        │             320 │
│ (DepthwiseConv2D)                    │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_2                │ (None, 112, 112, 32)        │             128 │
│ (BatchNormalization)                 │                             │                 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 112, 112, 64)        │           2,112 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ batch_normalization_3                │ (None, 112, 112, 64)        │             256 │
```

```
| (BatchNormalization)              |                           |
|
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| max_pooling2d_1 (MaxPooling2D)  | (None, 56, 56, 64)       |
0 |
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| depthwise_conv2d_1               | (None, 56, 56, 64)       |
640 |
| (DepthwiseConv2D)                |                           |
|
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| batch_normalization_4            | (None, 56, 56, 64)       |
256 |
| (BatchNormalization)             |                           |
|
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| conv2d_2 (Conv2D)                | (None, 56, 56, 128)      |
8,320 |
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| batch_normalization_5            | (None, 56, 56, 128)      |
512 |
| (BatchNormalization)             |                           |
|
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| max_pooling2d_2 (MaxPooling2D)  | (None, 28, 28, 128)      |
0 |
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| depthwise_conv2d_2               | (None, 28, 28, 128)      |
1,280 |
| (DepthwiseConv2D)                |                           |
|
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| batch_normalization_6            | (None, 28, 28, 128)      |
512 |
| (BatchNormalization)             |                           |
|
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
| conv2d_3 (Conv2D)                | (None, 28, 28, 256)      |
33,024 |
├──────────────────────────────────┼──────────────────────────┼──────
──────┤
```

```
| batch_normalization_7            | (None, 28, 28, 256)    |
1,024 |
| (BatchNormalization)             |                        |
|
├──────────────────────────────────┼────────────────────────┼──────────
──────┤
| max_pooling2d_3 (MaxPooling2D)   | (None, 14, 14, 256)    |
0 |
├──────────────────────────────────┼────────────────────────┼──────────
──────┤
| global_average_pooling2d         | (None, 256)            |
0 |
| (GlobalAveragePooling2D)         |                        |
|
├──────────────────────────────────┼────────────────────────┼──────────
──────┤
| dense (Dense)                    | (None, 512)            |
131,584 |
├──────────────────────────────────┼────────────────────────┼──────────
──────┤
| dropout (Dropout)                | (None, 512)            |
0 |
├──────────────────────────────────┼────────────────────────┼──────────
──────┤
| dense_1 (Dense)                  | (None, 3)              |
1,539 |
└──────────────────────────────────┴────────────────────────┴──────────
──────┘

 Total params: 182,974 (714.74 KB)

 Trainable params: 181,502 (708.99 KB)

 Non-trainable params: 1,472 (5.75 KB)


early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=15,
    restore_best_weights=True
)

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,
patience=3, min_lr=1e-7)

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // 64,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // 64,
```

```
    callbacks=[early_stopping, reduce_lr]
)

C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
c:\ProgramData\anaconda3\Lib\site-packages\PIL\Image.py:1000:
UserWarning: Palette images with Transparency expressed in bytes
should be converted to RGBA images
  warnings.warn(

Epoch 1/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 23s 5s/step - accuracy: 0.2958 - loss:
12.9285 - val_accuracy: 0.3333 - val_loss: 11.6155 - learning_rate:
1.0000e-04
Epoch 2/50
1/3 ━━━━━━━━━━━━━━━━━━━━ 5s 3s/step - accuracy: 0.3125 - loss: 12.6263

c:\ProgramData\anaconda3\Lib\contextlib.py:158: UserWarning: Your
input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs`
batches. You may need to use the `.repeat()` function when building
your dataset.
  self.gen.throw(value)

3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 418ms/step - accuracy: 0.3125 - loss:
12.6263 - val_accuracy: 0.3333 - val_loss: 11.6076 - learning_rate:
1.0000e-04
Epoch 3/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.2569 - loss:
12.3980 - val_accuracy: 0.3333 - val_loss: 11.5847 - learning_rate:
1.0000e-04
Epoch 4/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 373ms/step - accuracy: 0.3281 - loss:
12.0828 - val_accuracy: 0.3333 - val_loss: 11.5770 - learning_rate:
1.0000e-04
Epoch 5/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.3846 - loss:
12.1493 - val_accuracy: 0.3333 - val_loss: 11.5530 - learning_rate:
1.0000e-04
Epoch 6/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 371ms/step - accuracy: 0.3438 - loss:
12.2243 - val_accuracy: 0.3333 - val_loss: 11.5449 - learning_rate:
1.0000e-04
Epoch 7/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4116 - loss:
```

```
12.1452 - val_accuracy: 0.3333 - val_loss: 11.5199 - learning_rate:
1.0000e-04
Epoch 8/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 381ms/step - accuracy: 0.4375 - loss:
11.9481 - val_accuracy: 0.3333 - val_loss: 11.5115 - learning_rate:
1.0000e-04
Epoch 9/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.3282 - loss:
12.1297 - val_accuracy: 0.3333 - val_loss: 11.4858 - learning_rate:
1.0000e-04
Epoch 10/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 373ms/step - accuracy: 0.4062 - loss:
11.9944 - val_accuracy: 0.3333 - val_loss: 11.4771 - learning_rate:
1.0000e-04
Epoch 11/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.3778 - loss:
11.9597 - val_accuracy: 0.3333 - val_loss: 11.4510 - learning_rate:
1.0000e-04
Epoch 12/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 378ms/step - accuracy: 0.3594 - loss:
11.8576 - val_accuracy: 0.3333 - val_loss: 11.4422 - learning_rate:
1.0000e-04
Epoch 13/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 13s 4s/step - accuracy: 0.4655 - loss:
11.6413 - val_accuracy: 0.3333 - val_loss: 11.4155 - learning_rate:
1.0000e-04
Epoch 14/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 378ms/step - accuracy: 0.6042 - loss:
11.3457 - val_accuracy: 0.3333 - val_loss: 11.4065 - learning_rate:
1.0000e-04
Epoch 15/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4372 - loss:
11.6762 - val_accuracy: 0.3333 - val_loss: 11.3794 - learning_rate:
1.0000e-04
Epoch 16/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 387ms/step - accuracy: 0.4062 - loss:
11.7486 - val_accuracy: 0.3333 - val_loss: 11.3704 - learning_rate:
1.0000e-04
Epoch 17/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 13s 4s/step - accuracy: 0.4811 - loss:
11.6601 - val_accuracy: 0.3333 - val_loss: 11.3431 - learning_rate:
1.0000e-04
Epoch 18/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 379ms/step - accuracy: 0.4583 - loss:
11.4606 - val_accuracy: 0.3333 - val_loss: 11.3340 - learning_rate:
1.0000e-04
Epoch 19/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 13s 4s/step - accuracy: 0.4298 - loss:
11.6294 - val_accuracy: 0.3333 - val_loss: 11.3065 - learning_rate:
```

```
1.0000e-04
Epoch 20/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 375ms/step - accuracy: 0.5156 - loss:
11.4933 - val_accuracy: 0.3333 - val_loss: 11.2973 - learning_rate:
1.0000e-04
Epoch 21/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4629 - loss:
11.5428 - val_accuracy: 0.3333 - val_loss: 11.2699 - learning_rate:
1.0000e-04
Epoch 22/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 380ms/step - accuracy: 0.4531 - loss:
11.6066 - val_accuracy: 0.3333 - val_loss: 11.2607 - learning_rate:
1.0000e-04
Epoch 23/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4662 - loss:
11.5135 - val_accuracy: 0.3333 - val_loss: 11.2331 - learning_rate:
1.0000e-04
Epoch 24/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 390ms/step - accuracy: 0.4062 - loss:
11.6785 - val_accuracy: 0.3333 - val_loss: 11.2238 - learning_rate:
1.0000e-04
Epoch 25/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.3623 - loss:
11.4908 - val_accuracy: 0.3333 - val_loss: 11.1961 - learning_rate:
1.0000e-04
Epoch 26/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 377ms/step - accuracy: 0.3750 - loss:
11.5956 - val_accuracy: 0.3333 - val_loss: 11.1869 - learning_rate:
1.0000e-04
Epoch 27/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4945 - loss:
11.2430 - val_accuracy: 0.3333 - val_loss: 11.1594 - learning_rate:
1.0000e-04
Epoch 28/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 379ms/step - accuracy: 0.4375 - loss:
11.5838 - val_accuracy: 0.3333 - val_loss: 11.1502 - learning_rate:
1.0000e-04
Epoch 29/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4288 - loss:
11.4869 - val_accuracy: 0.3333 - val_loss: 11.1225 - learning_rate:
1.0000e-04
Epoch 30/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 378ms/step - accuracy: 0.3125 - loss:
11.4887 - val_accuracy: 0.3333 - val_loss: 11.1133 - learning_rate:
1.0000e-04
Epoch 31/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4219 - loss:
11.3775 - val_accuracy: 0.3333 - val_loss: 11.0860 - learning_rate:
1.0000e-04
```

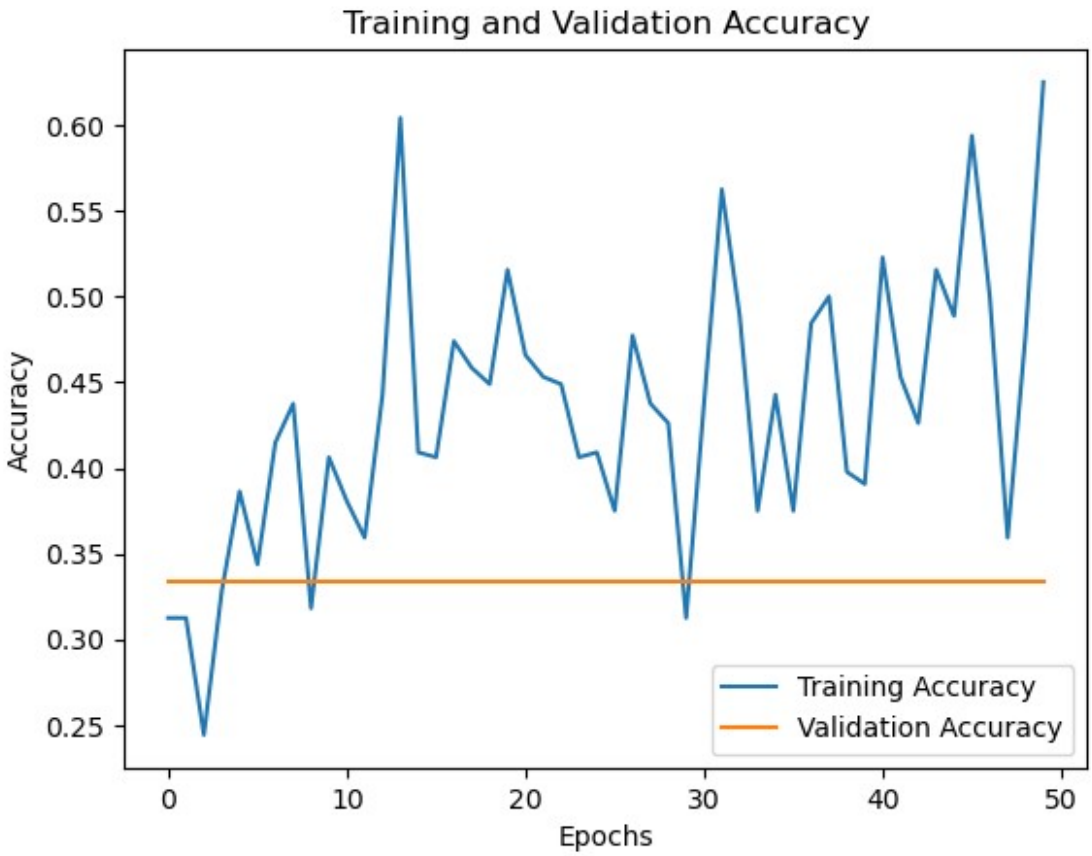
```
Epoch 32/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 383ms/step - accuracy: 0.5625 - loss:
10.9782 - val_accuracy: 0.3333 - val_loss: 11.0771 - learning_rate:
1.0000e-04
Epoch 33/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4787 - loss:
11.3559 - val_accuracy: 0.3333 - val_loss: 11.0502 - learning_rate:
1.0000e-04
Epoch 34/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 377ms/step - accuracy: 0.3750 - loss:
11.2711 - val_accuracy: 0.3333 - val_loss: 11.0413 - learning_rate:
1.0000e-04
Epoch 35/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 14s 4s/step - accuracy: 0.4225 - loss:
11.2514 - val_accuracy: 0.3333 - val_loss: 11.0142 - learning_rate:
1.0000e-04
Epoch 36/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 379ms/step - accuracy: 0.3750 - loss:
11.3842 - val_accuracy: 0.3333 - val_loss: 11.0053 - learning_rate:
1.0000e-04
Epoch 37/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 13s 4s/step - accuracy: 0.4609 - loss:
11.2846 - val_accuracy: 0.3333 - val_loss: 10.9784 - learning_rate:
1.0000e-04
Epoch 38/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 3s 380ms/step - accuracy: 0.5000 - loss:
10.9056 - val_accuracy: 0.3333 - val_loss: 10.9694 - learning_rate:
1.0000e-04
Epoch 39/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.3864 - loss:
11.1736 - val_accuracy: 0.3333 - val_loss: 10.9424 - learning_rate:
1.0000e-04
Epoch 40/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 378ms/step - accuracy: 0.3906 - loss:
11.0615 - val_accuracy: 0.3333 - val_loss: 10.9333 - learning_rate:
1.0000e-04
Epoch 41/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.5368 - loss:
10.9183 - val_accuracy: 0.3333 - val_loss: 10.9064 - learning_rate:
1.0000e-04
Epoch 42/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 380ms/step - accuracy: 0.4531 - loss:
11.1512 - val_accuracy: 0.3333 - val_loss: 10.8975 - learning_rate:
1.0000e-04
Epoch 43/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4377 - loss:
10.9906 - val_accuracy: 0.3333 - val_loss: 10.8712 - learning_rate:
1.0000e-04
Epoch 44/50
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 380ms/step - accuracy: 0.5156 - loss:
10.9416 - val_accuracy: 0.3333 - val_loss: 10.8625 - learning_rate:
1.0000e-04
Epoch 45/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step - accuracy: 0.4859 - loss:
10.9074 - val_accuracy: 0.3333 - val_loss: 10.8357 - learning_rate:
1.0000e-04
Epoch 46/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 377ms/step - accuracy: 0.5938 - loss:
10.7015 - val_accuracy: 0.3333 - val_loss: 10.8269 - learning_rate:
1.0000e-04
Epoch 47/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 5s/step - accuracy: 0.4896 - loss:
10.9206 - val_accuracy: 0.3333 - val_loss: 10.8003 - learning_rate:
1.0000e-04
Epoch 48/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 414ms/step - accuracy: 0.3594 - loss:
10.9371 - val_accuracy: 0.3333 - val_loss: 10.7916 - learning_rate:
1.0000e-04
Epoch 49/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 13s 5s/step - accuracy: 0.4685 - loss:
10.8150 - val_accuracy: 0.3333 - val_loss: 10.7658 - learning_rate:
1.0000e-04
Epoch 50/50
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 378ms/step - accuracy: 0.6250 - loss:
10.5879 - val_accuracy: 0.3333 - val_loss: 10.7571 - learning_rate:
1.0000e-04
```

```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 732ms/step - accuracy: 0.3333 - loss:
10.7570
Test Loss: 10.75701904296875
Test Accuracy: 0.3333333432674408
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
# Plot Loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

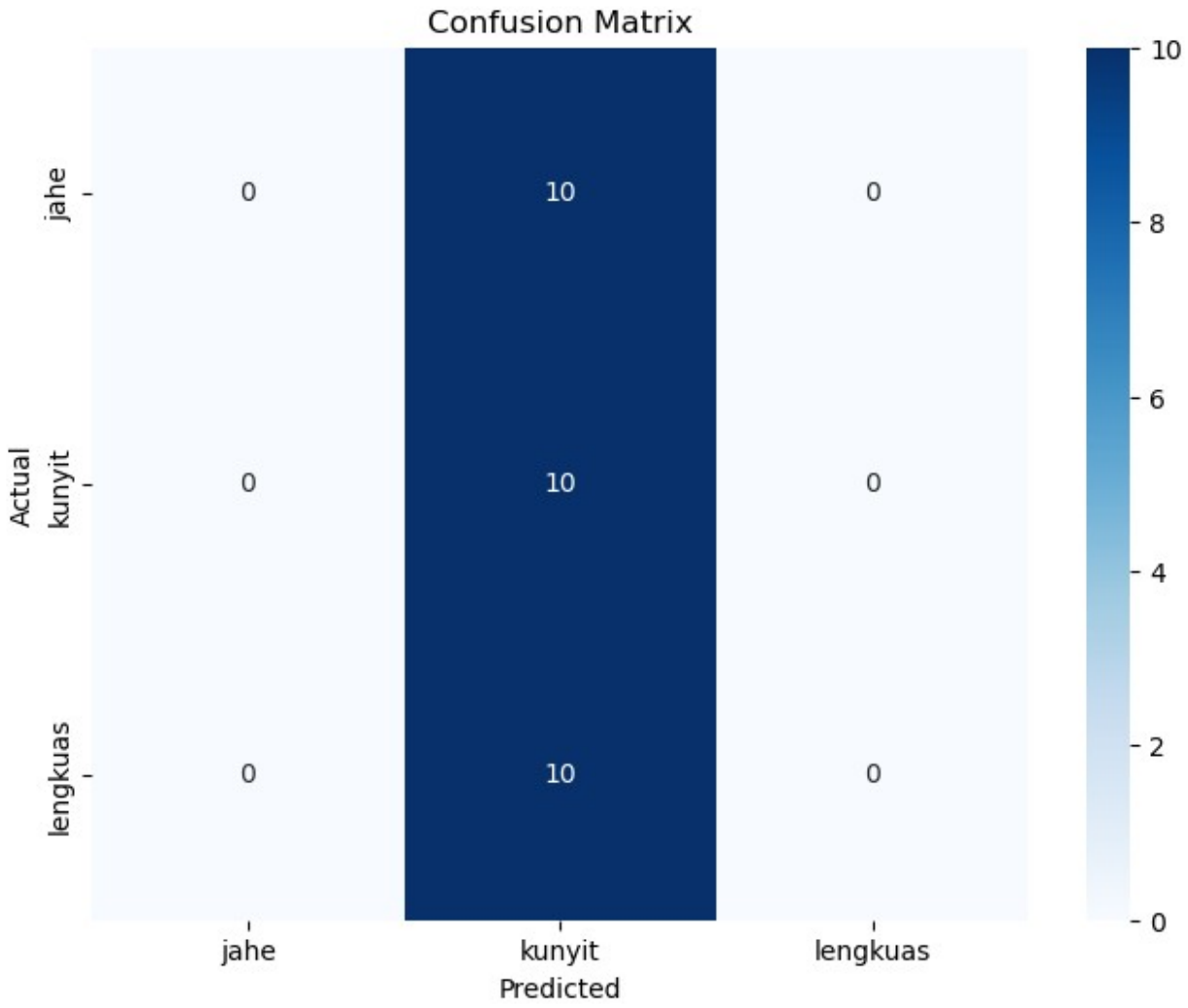

```
model.save('BestModel_MobileNet_AlbertCornelius.h5')
```

```
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')`.
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

test_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_generator = test_datagen.flow_from_directory(
    r'test',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)
```

```
model = load_model('BestModel_MobileNet_AlbertCornelius.h5')

test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")

y_pred = np.argmax(model.predict(test_generator), axis=1)
y_true = test_generator.classes

conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=test_generator.class_indices.keys(),
            yticklabels=test_generator.class_indices.keys())
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

class_report = classification_report(y_true, y_pred,
target_names=test_generator.class_indices.keys())
print("Classification Report:")
print(class_report)
```

```
Found 30 images belonging to 3 classes.

WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.3333 - loss: 10.7570
Test Loss: 10.75701904296875
Test Accuracy: 0.3333333432674408
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 996ms/step
```




```
Classification Report:
              precision    recall  f1-score   support

        jahe       0.00      0.00      0.00        10
      kunyit       0.33      1.00      0.50        10
    lengkuas       0.00      0.00      0.00        10

    accuracy                           0.33        30
   macro avg       0.11      0.33      0.17        30
weighted avg       0.11      0.33      0.17        30


c:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
c:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
import os
import numpy as np

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

count = 0
dirs = os.listdir(r'C:\Users\ASUS\Documents\dataset\Google net\train')
for i in dirs:
    files = list(os.listdir(r'C:\Users\ASUS\Documents\dataset\Google
net\train' + '/' + i))
    count += len(files)
    print(i + ' Folder has ' + str(len(files)) + ' images')
print('Total images in train folder: ', count)
```

```
jahe Folder has 100 images
kunyit Folder has 100 images
lengkuas Folder has 100 images
Total images in train folder:  300
```

```
import os
import shutil
from sklearn.model_selection import train_test_split

base_dir = r"C:\Users\ASUS\Documents\dataset\Google net\train"
validation_dir = r"C:\Users\ASUS\Documents\dataset\Google net\
validation"
test_dir = r"C:\Users\ASUS\Documents\dataset\Google net\test"

os.makedirs(validation_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)

categories = [name for name in os.listdir(base_dir) if
os.path.isdir(os.path.join(base_dir, name))]

validation_split = 0.1  # 10% untuk validasi
test_split = 0.1        # 10% untuk test
train_split = 1 - validation_split - test_split  # 80% untuk train

for category in categories:
    category_dir = os.path.join(base_dir, category)
    validation_category_dir = os.path.join(validation_dir, category)
    test_category_dir = os.path.join(test_dir, category)

    os.makedirs(validation_category_dir, exist_ok=True)
    os.makedirs(test_category_dir, exist_ok=True)
```

```
    all_files = os.listdir(category_dir)

    train_val_files, test_files = train_test_split(all_files,
test_size=test_split, random_state=42)

    train_files, val_files = train_test_split(train_val_files,
test_size=validation_split / (train_split + validation_split),
random_state=42)

    for file in val_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(validation_category_dir, file))

    for file in test_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(test_category_dir, file))

print("Dataset berhasil dibagi menjadi train, validation, dan test!")
```

```
Dataset berhasil dibagi menjadi train, validation, dan test!
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)

validation_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
    r'train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

```
validation_generator = validation_datagen.flow_from_directory(
    r'validation',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    r'test',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 240 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
```

```
sample_batch = next(train_generator)

plt.figure(figsize=(10, 10))
for i in range(min(9, len(sample_batch[0]))):
    image = sample_batch[0][i]
    label = sample_batch[1][i]
    plt.subplot(3, 3, i + 1)
    plt.imshow(image)
    plt.title(f"Label: {label}\nSize: {image.shape}")
    plt.axis('off')
plt.show()
```


```
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
AveragePooling2D, Flatten, Dense, Dropout, Input, concatenate
from tensorflow.keras.models import Model

def inception_block(x, filters):
    conv_1x1 = Conv2D(filters[0], (1, 1), padding='same',
activation='relu')(x)

    conv_3x3_reduce = Conv2D(filters[1], (1, 1), padding='same',
activation='relu')(x)
    conv_3x3 = Conv2D(filters[2], (3, 3), padding='same',
```

```
activation='relu')(conv_3x3_reduce)

    conv_5x5_reduce = Conv2D(filters[3], (1, 1), padding='same',
activation='relu')(x)
    conv_5x5 = Conv2D(filters[4], (5, 5), padding='same',
activation='relu')(conv_5x5_reduce)

    pool_proj = MaxPooling2D((3, 3), strides=(1, 1), padding='same')
(x)
    pool_proj = Conv2D(filters[5], (1, 1), padding='same',
activation='relu')(pool_proj)

    output = concatenate([conv_1x1, conv_3x3, conv_5x5, pool_proj],
axis=-1)

    return output

def googlenet(input_shape, num_classes):
    input_layer = Input(shape=input_shape)

    x = Conv2D(64, (7, 7), strides=(2, 2), padding='same',
activation='relu')(input_layer)
    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)

    x = inception_block(x, [64, 96, 128, 16, 32, 32])
    x = inception_block(x, [128, 128, 192, 32, 96, 64])
    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)

    x = inception_block(x, [192, 96, 208, 16, 48, 64])
    x = inception_block(x, [256, 160, 320, 32, 128, 128])

    x = AveragePooling2D((7, 7), strides=(1, 1))(x)
    x = Flatten()(x)
    x = Dropout(0.4)(x)
    output_layer = Dense(num_classes, activation='softmax')(x)

    model = Model(input_layer, output_layer)
    return model

model = googlenet((224, 224, 3), num_classes=3)
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

history = model.fit(train_generator,
                    validation_data=validation_generator,
```

```
                    epochs=50,
                    callbacks=[early_stopping])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

Epoch 1/50

c:\ProgramData\anaconda3\Lib\site-packages\PIL\Image.py:1000:
UserWarning: Palette images with Transparency expressed in bytes
should be converted to RGBA images
  warnings.warn(

8/8 ━━━━━━━━━━━━━━━━━━━━ 29s 2s/step - accuracy: 0.3866 - loss: 1.8479
- val_accuracy: 0.4333 - val_loss: 1.1568
Epoch 2/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.3892 - loss: 1.1270
- val_accuracy: 0.5333 - val_loss: 0.9242
Epoch 3/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.5468 - loss: 0.9119
- val_accuracy: 0.4667 - val_loss: 1.1613
Epoch 4/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.5073 - loss: 0.9402
- val_accuracy: 0.5667 - val_loss: 0.7686
Epoch 5/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.5826 - loss: 0.8259
- val_accuracy: 0.6333 - val_loss: 0.6487
Epoch 6/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.6078 - loss: 0.8279
- val_accuracy: 0.6333 - val_loss: 0.6441
Epoch 7/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.6096 - loss: 0.8026
- val_accuracy: 0.6333 - val_loss: 0.6737
Epoch 8/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.5422 - loss: 0.7729
- val_accuracy: 0.6333 - val_loss: 0.6276
Epoch 9/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.5762 - loss: 0.8955
- val_accuracy: 0.3667 - val_loss: 1.0968
Epoch 10/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.5164 - loss: 1.0678
- val_accuracy: 0.6000 - val_loss: 0.7976
Epoch 11/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.6077 - loss: 0.8601
- val_accuracy: 0.6333 - val_loss: 0.6379
```

```
Epoch 12/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.6183 - loss: 0.7398
- val_accuracy: 0.6333 - val_loss: 0.7139
Epoch 13/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.6578 - loss: 0.8337
- val_accuracy: 0.6000 - val_loss: 0.7131
```

```
test_loss, test_acc = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_acc}")
```

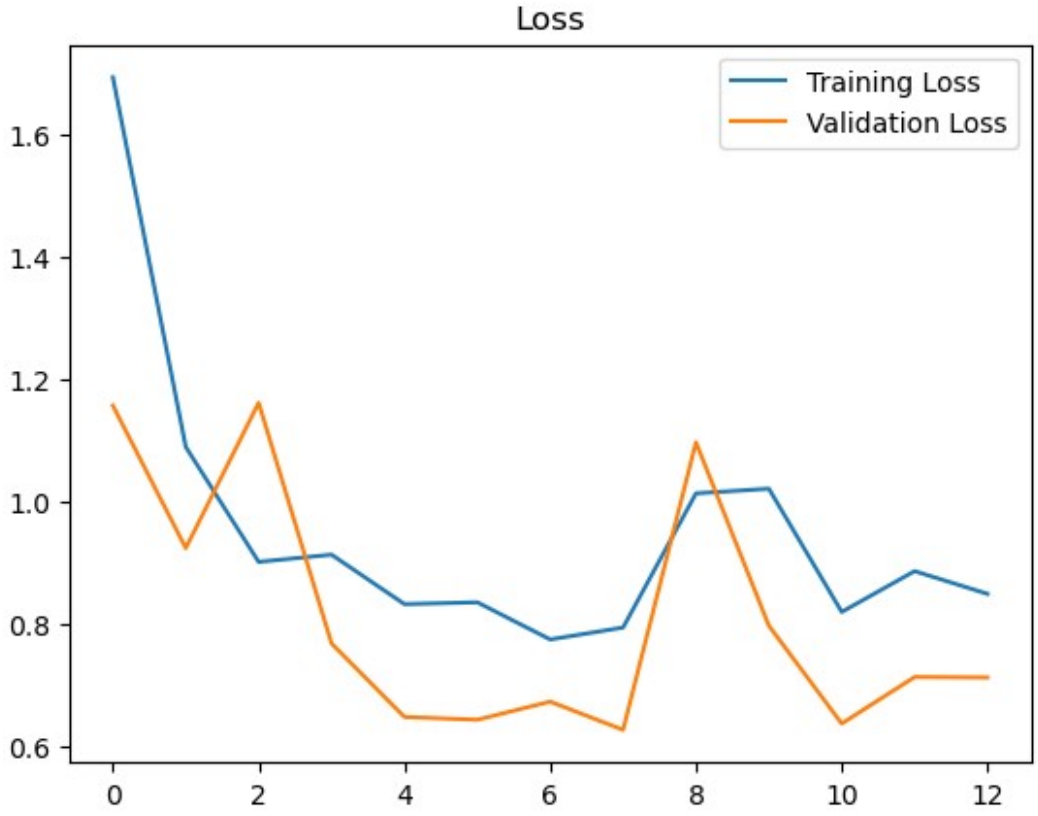
```
import matplotlib.pyplot as plt
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title("Accuracy")
plt.show()
```

```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title("Loss")
plt.show()
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.6000 - loss: 0.7099
Test Loss: 0.709863007068634
Test Accuracy: 0.6000000238418579
```


Accuracy
0.65
0.60
0.55
0.50
0.45
0.40
0
2
4
6
8
10
12
Training Accuracy
Validation Accuracy


Loss
1.6
1.4
1.2
1.0
0.8
0.6
0
2
4
6
8
10
12
Training Loss
Validation Loss

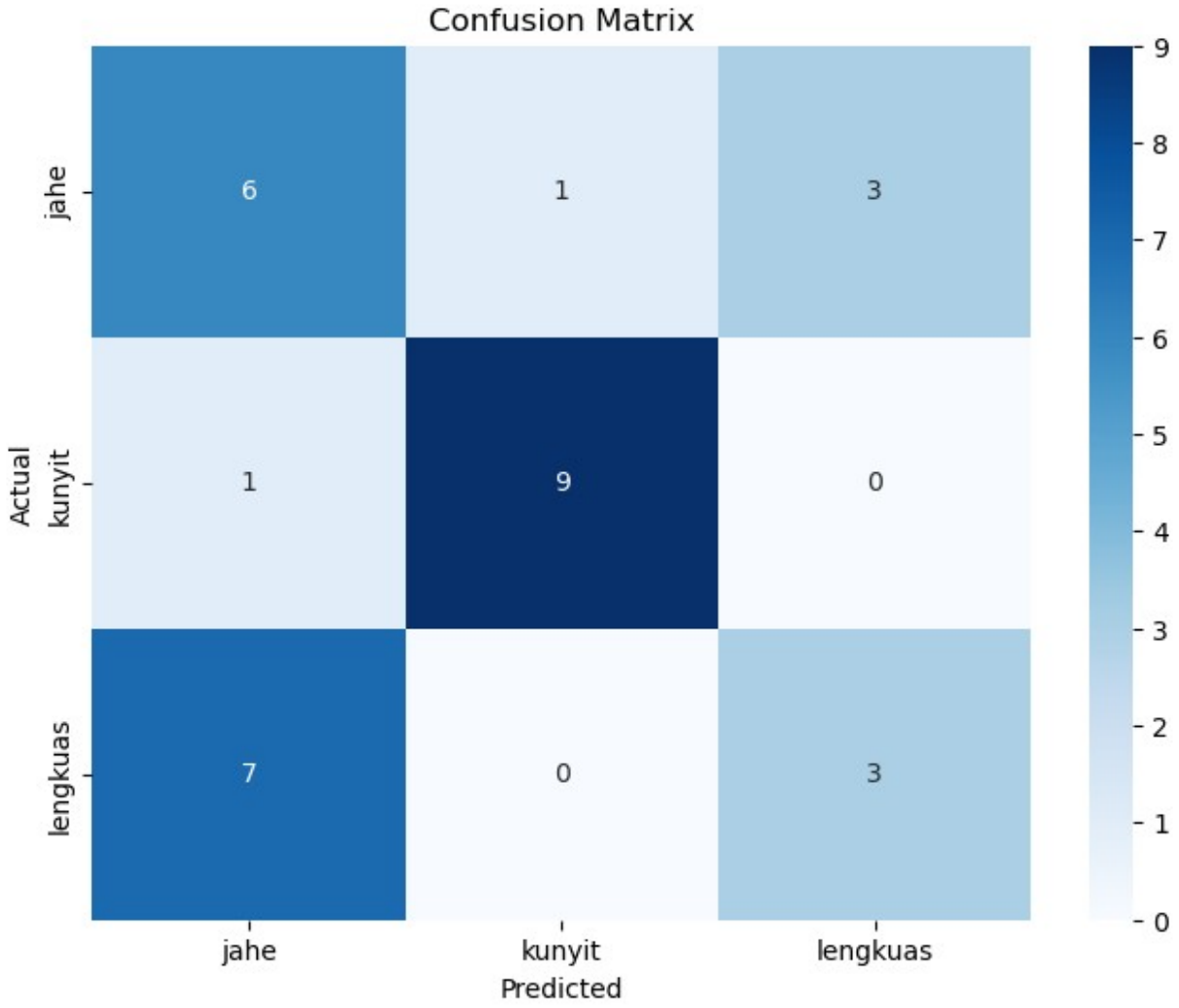
```
model.save('googleNet_model.h5')
```

```
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')`.
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

test_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_generator = test_datagen.flow_from_directory(
    'test',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

model = load_model('googleNet_model.h5')

test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")

y_pred = np.argmax(model.predict(test_generator), axis=1)
y_true = test_generator.classes

conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=test_generator.class_indices.keys(),
            yticklabels=test_generator.class_indices.keys())
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

class_report = classification_report(y_true, y_pred,
target_names=test_generator.class_indices.keys())
print("Classification Report:")
print(class_report)
```

```
Found 30 images belonging to 3 classes.
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.6000 - loss: 0.7099
Test Loss: 0.709863007068634
Test Accuracy: 0.6000000238418579
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
```



```
Classification Report:
              precision    recall  f1-score   support

        jahe       0.43      0.60      0.50        10
      kunyit       0.90      0.90      0.90        10
    lengkuas       0.50      0.30      0.38        10

    accuracy                           0.60        30
   macro avg       0.61      0.60      0.59        30
weighted avg       0.61      0.60      0.59        30
```

```
import os
import numpy as np

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

count = 0
dirs = os.listdir(r'C:\Users\ASUS\Documents\dataset\dataset train
test\train')
for i in dirs:
    files = list(os.listdir(r'C:\Users\ASUS\Documents\dataset\dataset
train test\train' + '/' + i))
    count += len(files)
    print(i + ' punya ' + str(len(files)) + ' images')
print('Total gambar dalam folder: ', count)
```

```
jahe punya 80 images
kunyit punya 80 images
lengkuas punya 80 images
Total gambar dalam folder:  240
```

```
import os
import shutil
from sklearn.model_selection import train_test_split

base_dir = r"C:\Users\ASUS\Documents\dataset\dataset train test\train"
validation_dir = r"C:\Users\ASUS\Documents\dataset\dataset train test\
validation"
test_dir = r"C:\Users\ASUS\Documents\dataset\dataset train test\test"

os.makedirs(validation_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)

categories = [name for name in os.listdir(base_dir) if
os.path.isdir(os.path.join(base_dir, name))]

validation_split = 0.1  # 10% untuk validasi
test_split = 0.1        # 10% untuk test
train_split = 1 - validation_split - test_split  # 80% untuk train

for category in categories:
    category_dir = os.path.join(base_dir, category)
    validation_category_dir = os.path.join(validation_dir, category)
    test_category_dir = os.path.join(test_dir, category)

    os.makedirs(validation_category_dir, exist_ok=True)
```

```
    os.makedirs(test_category_dir, exist_ok=True)

    all_files = os.listdir(category_dir)

    train_val_files, test_files = train_test_split(all_files,
test_size=test_split, random_state=42)

    train_files, val_files = train_test_split(train_val_files,
test_size=validation_split / (train_split + validation_split),
random_state=42)

    for file in val_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(validation_category_dir, file))

    for file in test_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(test_category_dir, file))

print("Dataset berhasil dibagi menjadi train, validation, dan test!")
```

```
Dataset berhasil dibagi menjadi train, validation, dan test!
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)

validation_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
    r'dataset train test\train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
```

```
)

validation_generator = validation_datagen.flow_from_directory(
    r'dataset train test\validation',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    r'dataset train test\test',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 240 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
```

```
Found 30 images belonging to 3 classes.
```

```
sample_batch = next(train_generator)

plt.figure(figsize=(10, 10))
for i in range(min(9, len(sample_batch[0]))):
    image = sample_batch[0][i]
    label = sample_batch[1][i]
    plt.subplot(3, 3, i + 1)
    plt.imshow(image)
    plt.title(f"Label: {label}\nSize: {image.shape}")
    plt.axis('off')
plt.show()
```

```
c:\ProgramData\anaconda3\Lib\site-packages\PIL\Image.py:1000:
UserWarning: Palette images with Transparency expressed in bytes
should be converted to RGBA images
  warnings.warn(
```


```
model = Sequential([
    Conv2D(64, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D(pool_size=(2, 2)),

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

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

    Conv2D(512, (3, 3), activation='relu'),
```

```
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')
])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

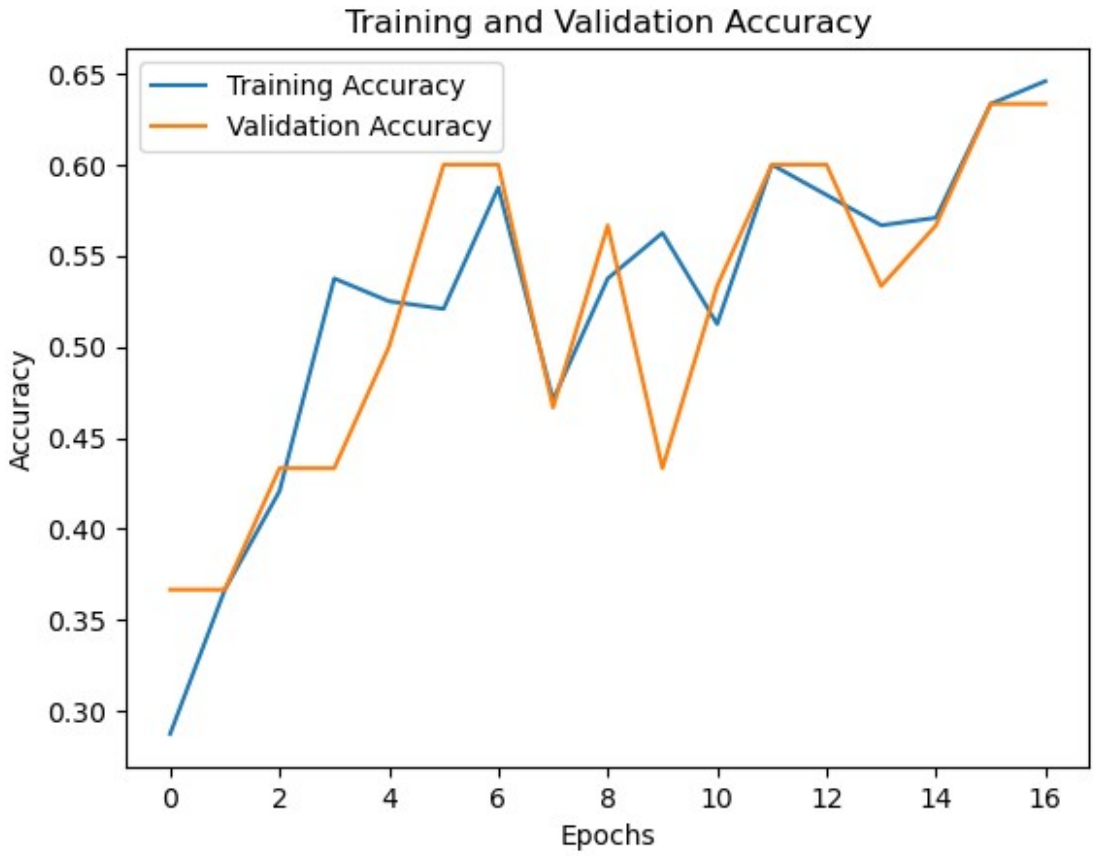
```
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)


early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

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

```
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()

Epoch 1/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 21s 2s/step - accuracy: 0.2794 - loss: 1.1428 - val_accuracy: 0.3667 - val_loss: 1.0979
Epoch 2/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.3753 - loss: 1.0940 - val_accuracy: 0.3667 - val_loss: 1.0821
Epoch 3/50
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.4518 - loss: 1.0715
- val_accuracy: 0.4333 - val_loss: 1.0479
Epoch 4/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.5021 - loss: 1.0233
- val_accuracy: 0.4333 - val_loss: 0.9929
Epoch 5/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.5511 - loss: 0.9399
- val_accuracy: 0.5000 - val_loss: 0.9147
Epoch 6/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.5388 - loss: 0.8841
- val_accuracy: 0.6000 - val_loss: 0.8258
Epoch 7/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.5773 - loss: 0.8721
- val_accuracy: 0.6000 - val_loss: 0.7915
Epoch 8/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.4828 - loss: 0.9472
- val_accuracy: 0.4667 - val_loss: 0.8554
Epoch 9/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.5175 - loss: 0.8999
- val_accuracy: 0.5667 - val_loss: 0.7401
Epoch 10/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.5779 - loss: 0.8129
- val_accuracy: 0.4333 - val_loss: 0.7636
Epoch 11/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.5090 - loss: 0.8052
- val_accuracy: 0.5333 - val_loss: 0.7419
Epoch 12/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.6187 - loss: 0.8045
- val_accuracy: 0.6000 - val_loss: 0.6577
Epoch 13/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 0.5695 - loss: 0.8565
- val_accuracy: 0.6000 - val_loss: 0.6785
Epoch 14/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 20s 2s/step - accuracy: 0.5695 - loss: 0.8504
- val_accuracy: 0.5333 - val_loss: 0.7693
Epoch 15/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 17s 2s/step - accuracy: 0.5672 - loss: 0.7929
- val_accuracy: 0.5667 - val_loss: 0.6895
Epoch 16/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.6633 - loss: 0.7523
- val_accuracy: 0.6333 - val_loss: 0.7073
Epoch 17/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.6890 - loss: 0.7040
- val_accuracy: 0.6333 - val_loss: 0.7106

test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 927ms/step - accuracy: 0.6667 - loss:
0.7331
Test Loss: 0.7331267595291138
Test Accuracy: 0.6666666865348816
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

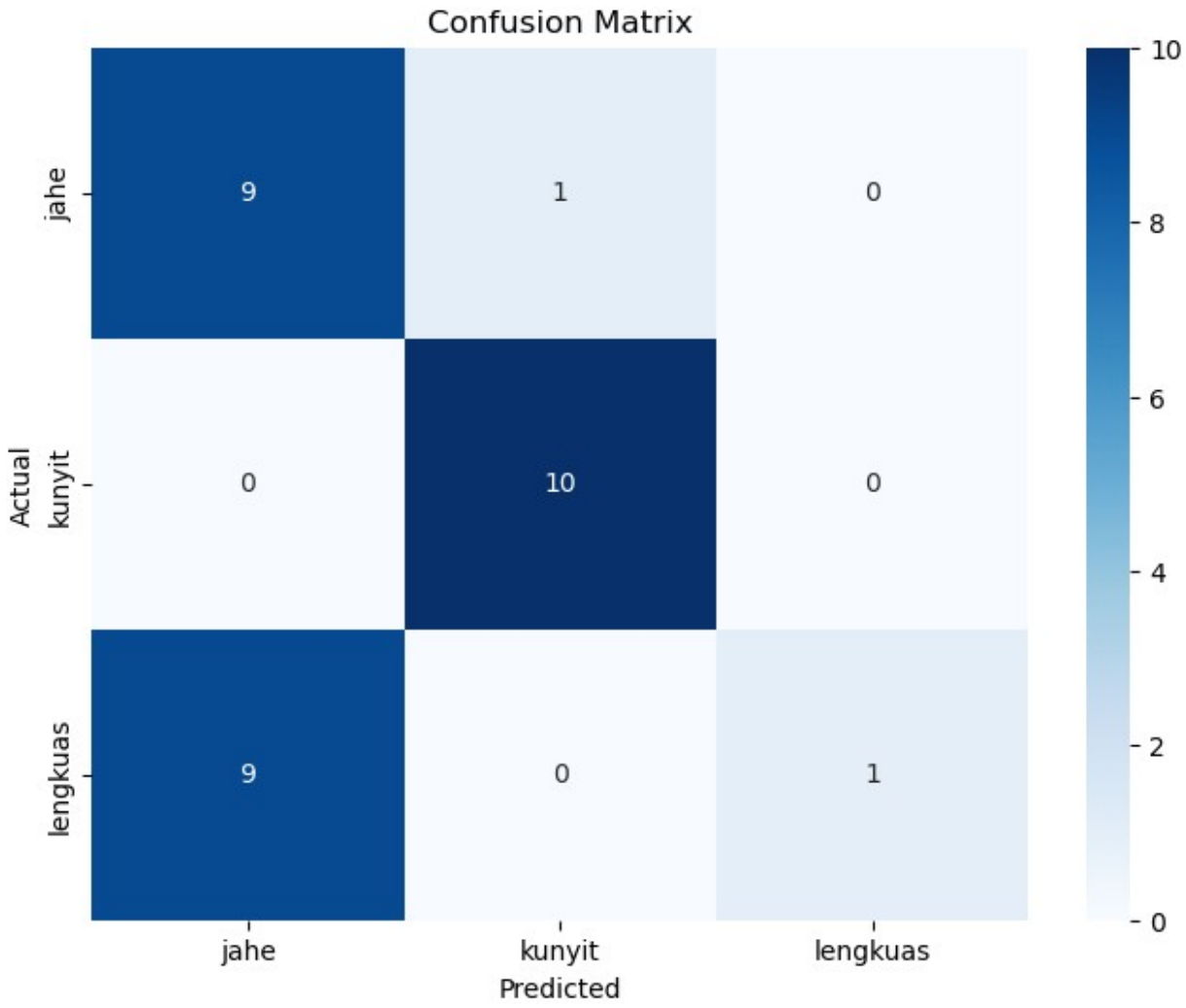

```
model.save('VGG rev.h5')
```

```
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')`.
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

test_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_generator = test_datagen.flow_from_directory(
    'dataset train test/test',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)
```

```
model = load_model('VGG rev.h5')

test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")

y_pred = np.argmax(model.predict(test_generator), axis=1)
y_true = test_generator.classes

conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=test_generator.class_indices.keys(),
            yticklabels=test_generator.class_indices.keys())
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Classification Report
class_report = classification_report(y_true, y_pred,
target_names=test_generator.class_indices.keys())
print("Classification Report:")
print(class_report)
```

```
Found 30 images belonging to 3 classes.

WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.6667 - loss: 0.7331
Test Loss: 0.7331267595291138
Test Accuracy: 0.6666666865348816
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 996ms/step
```




```
Classification Report:
              precision    recall  f1-score   support

       jahe       0.50      0.90      0.64        10
     kunyit       0.91      1.00      0.95        10
   lengkuas       1.00      0.10      0.18        10

   accuracy                           0.67        30
  macro avg       0.80      0.67      0.59        30
weighted avg       0.80      0.67      0.59        30
```

```
import os
import numpy as np

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

count = 0
dirs = os.listdir(r'C:\Users\ASUS\Documents\dataset\Alex net\train')
for i in dirs:
    files = list(os.listdir(r'C:\Users\ASUS\Documents\dataset\Alex
net\train' + '/' + i))
    count += len(files)
    print(i + ' punya ' + str(len(files)) + ' images')
print('Total gambar dalam folder: ', count)
```

```
jahe punya 100 images
kunyit punya 100 images
lengkuas punya 100 images
Total gambar dalam folder:  300
```

```
import os
import shutil
from sklearn.model_selection import train_test_split

base_dir = r"C:\Users\ASUS\Documents\dataset\Alex net\train"
validation_dir = r"C:\Users\ASUS\Documents\dataset\Alex net\
validation"
test_dir = r"C:\Users\ASUS\Documents\dataset\Alex net\test"

os.makedirs(validation_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)

categories = [name for name in os.listdir(base_dir) if
os.path.isdir(os.path.join(base_dir, name))]

validation_split = 0.1  # 10% untuk validasi
test_split = 0.1        # 10% untuk test
train_split = 1 - validation_split - test_split  # 80% untuk train

for category in categories:
    category_dir = os.path.join(base_dir, category)
    validation_category_dir = os.path.join(validation_dir, category)
    test_category_dir = os.path.join(test_dir, category)

    os.makedirs(validation_category_dir, exist_ok=True)
    os.makedirs(test_category_dir, exist_ok=True)
```

```
    all_files = os.listdir(category_dir)

    train_val_files, test_files = train_test_split(all_files,
test_size=test_split, random_state=42)

    train_files, val_files = train_test_split(train_val_files,
test_size=validation_split / (train_split + validation_split),
random_state=42)

    for file in val_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(validation_category_dir, file))

    for file in test_files:
        shutil.move(os.path.join(category_dir, file),
os.path.join(test_category_dir, file))

print("Dataset berhasil dibagi menjadi train, validation, dan test!")
```

```
Dataset berhasil dibagi menjadi train, validation, dan test!
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt

train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)

validation_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
    r"train",
    target_size=(227, 227),
    batch_size=32,
    class_mode='categorical'
)
```

```
validation_generator = validation_datagen.flow_from_directory(
    r'validation',
    target_size=(227, 227),
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    r'test',
    target_size=(227, 227),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 240 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
```

```
sample_batch = next(train_generator)

plt.figure(figsize=(10, 10))
for i in range(min(9, len(sample_batch[0]))):
    image = sample_batch[0][i]
    label = sample_batch[1][i]
    plt.subplot(3, 3, i + 1)
    plt.imshow(image)
    plt.title(f"Label: {label}\nSize: {image.shape}")
    plt.axis('off')
plt.show()
```


```
model = Sequential([
    Conv2D(96, (11, 11), strides=4, activation='relu',
input_shape=(227, 227, 3)),
    MaxPooling2D(pool_size=(3, 3), strides=2),

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

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

```
    MaxPooling2D(pool_size=(3, 3), strides=2),


    Flatten(),


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

    Dense(3, activation='softmax')
])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━
┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━
┩
│ conv2d (Conv2D)                 │ (None, 55, 55, 96)     │
34,944 │
├─────────────────────────────────┼────────────────────────┼───────────────
┤
│ max_pooling2d (MaxPooling2D)    │ (None, 27, 27, 96)     │
0 │
├─────────────────────────────────┼────────────────────────┼───────────────
┤
│ conv2d_1 (Conv2D)               │ (None, 27, 27, 256)    │
614,656 │
├─────────────────────────────────┼────────────────────────┼───────────────
┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 13, 13, 256)    │
```

```
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_2 (Conv2D)                   │ (None, 13, 13, 384)         │
885,120 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_3 (Conv2D)                   │ (None, 13, 13, 384)         │
1,327,488 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_4 (Conv2D)                   │ (None, 13, 13, 256)         │
884,992 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ max_pooling2d_2 (MaxPooling2D)      │ (None, 6, 6, 256)           │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ flatten (Flatten)                   │ (None, 9216)                │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense (Dense)                       │ (None, 4096)                │
37,752,832 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dropout (Dropout)                   │ (None, 4096)                │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_1 (Dense)                     │ (None, 4096)                │
16,781,312 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dropout_1 (Dropout)                 │ (None, 4096)                │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_2 (Dense)                     │ (None, 3)                   │
12,291 │
└─────────────────────────────────────┴─────────────────────────────┴──────────
──────┘

 Total params: 58,293,635 (222.37 MB)

 Trainable params: 58,293,635 (222.37 MB)

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

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

early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=4,
                               restore_best_weights=True)

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

```
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
c:\ProgramData\anaconda3\Lib\site-packages\PIL\Image.py:1000:
UserWarning: Palette images with Transparency expressed in bytes
should be converted to RGBA images
  warnings.warn(

Epoch 1/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.3164 - loss: 1.1078
- val_accuracy: 0.3333 - val_loss: 1.1013
Epoch 2/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.3872 - loss: 1.0827
- val_accuracy: 0.4667 - val_loss: 1.0513
Epoch 3/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.5037 - loss: 1.0088
- val_accuracy: 0.4333 - val_loss: 1.1171
Epoch 4/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.4726 - loss: 0.9442
- val_accuracy: 0.5667 - val_loss: 0.9019
Epoch 5/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.5817 - loss: 0.9011
- val_accuracy: 0.5000 - val_loss: 0.7816
Epoch 6/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.5403 - loss: 0.8573
- val_accuracy: 0.5000 - val_loss: 0.8945
Epoch 7/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.5775 - loss: 0.9364
- val_accuracy: 0.6333 - val_loss: 0.7763
Epoch 8/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.5939 - loss: 0.8170
- val_accuracy: 0.6000 - val_loss: 0.6924
Epoch 9/50
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 15s 2s/step - accuracy: 0.5393 - loss: 0.7739
- val_accuracy: 0.4667 - val_loss: 0.7426
Epoch 10/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 14s 2s/step - accuracy: 0.4952 - loss: 0.8215
- val_accuracy: 0.4333 - val_loss: 0.9822
Epoch 11/50
8/8 ━━━━━━━━━━━━━━━━━━━━ 14s 2s/step - accuracy: 0.5075 - loss: 0.7915
- val_accuracy: 0.6000 - val_loss: 0.6318
```

```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 782ms/step - accuracy: 0.6000 - loss:
0.7922
Test Loss: 0.7921562194824219
Test Accuracy: 0.6000000238418579
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```


```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

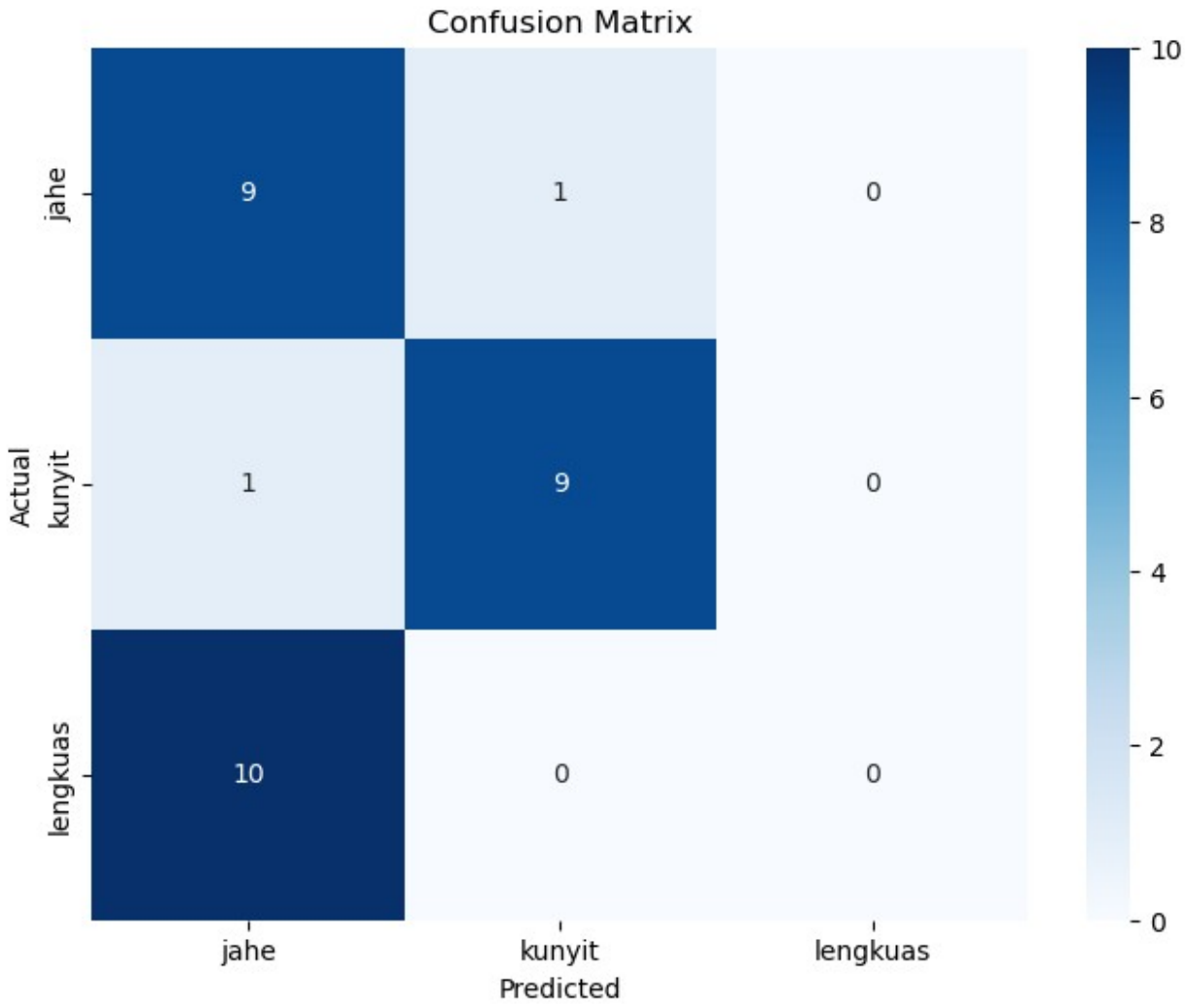

```
model.save('Alexnet.h5')
```

```
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')`.
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


test_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_generator = test_datagen.flow_from_directory(
    r'test',
    target_size=(227, 227),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
```

```
)

model = load_model('Alexnet.h5')


test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")

# Prediksi pada data test
y_pred = np.argmax(model.predict(test_generator), axis=1)
y_true = test_generator.classes

# Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=test_generator.class_indices.keys(),
            yticklabels=test_generator.class_indices.keys())
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Classification Report
class_report = classification_report(y_true, y_pred,
target_names=test_generator.class_indices.keys())
print("Classification Report:")
print(class_report)
```

```
Found 30 images belonging to 3 classes.

WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\keras\
src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its
constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.6000 - loss: 0.7922
Test Loss: 0.7921561598777771
Test Accuracy: 0.6000000238418579
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 816ms/step
```




```
Classification Report:
              precision    recall  f1-score   support

        jahe       0.45      0.90      0.60        10
      kunyit       0.90      0.90      0.90        10
    lengkuas       0.00      0.00      0.00        10

    accuracy                           0.60        30
   macro avg       0.45      0.60      0.50        30
weighted avg       0.45      0.60      0.50        30


c:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
c:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
c:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```