Flanella Joevita Vanling - 220711706

Kelompok H2O

Topik : Klasifikasi jenis jamur

VGG-16

In [1]:

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

data_dir = r"D:\Kuliah\SEM 5\ml\UAS\UASSSSS\datasetUASML\train_data"
data = tf.keras.utils.image_dataset_from_directory(data_dir, seed = 123, image_size=(224,224), batch_size=32)
print(data.class_names)

class_names = data.class_names
```

```
Found 300 files belonging to 3 classes.
['JamurKuping', 'JamurReishi', 'JamurShitake']
```

In [2]:

```
img_size = 224
batch = 32

dataset = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
)
```

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

In [3]:

```
train_split = 0.8
validation_split = 0.1
test_split = 0.1

total_count = len(dataset)

train_count = int(total_count * train_split)
val_count = int(total_count * validation_split)
test_count = total_count - train_count - val_count

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

train_ds = dataset.take(train_count)
remaining_ds = dataset.skip(train_count)

val_ds = remaining_ds.take(val_count)
test_ds = remaining_ds.skip(val_count)
```

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

In [4]:

```
import matplotlib.pyplot as plt

def visualize_images(dataset, class_names, num_images=9, img_size=(10, 10)):
    plt.figure(figsize=img_size)
    for images, labels in dataset.take(1):
        for i in range(num_images):
            plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype('uint8'))
            plt.title(class_names[labels[i]])
            plt.axis('off')
    plt.show()

visualize_images(train_ds, class_names, num_images=9, img_size=(10, 10))
```

In [5]:

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models

img_size = 224
batch = 32

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
    subset="training",
    validation_split=0.2
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
    subset="validation",
    validation_split=0.2
)

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

train_ds = train_ds.map(lambda x, y: (data_augmentation(x), y))

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

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

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

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

In [6]:

```
from tensorflow.keras import layers, models

input_layer = layers.Input(shape=(224, 224, 3))

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

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

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

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

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

x = layers.Flatten()(x)

x = layers.Dense(4096, activation='relu')(x)
x = layers.Dense(4096, activation='relu')(x)

output_layer = layers.Dense(3, activation='softmax')(x)

model = models.Model(inputs=input_layer, outputs=output_layer)

model.summary()
```

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
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 [ ]:

```
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_loss',
                               patience=5,
                               restore_best_weights=True)

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

```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 125s 14s/step - accuracy: 0.3044 - loss: 565.2374 - val_accuracy: 0.5167 -
val_loss: 1.0720
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 105s 13s/step - accuracy: 0.4164 - loss: 1.9042 - val_accuracy: 0.3500 - va
l_loss: 1.1041
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 100s 12s/step - accuracy: 0.2948 - loss: 1.1754 - val_accuracy: 0.3000 - va
l_loss: 1.0940
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 99s 12s/step - accuracy: 0.3489 - loss: 1.3574 - val_accuracy: 0.3500 - val
_loss: 1.1020
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 100s 12s/step - accuracy: 0.3049 - loss: 1.0996 - val_accuracy: 0.5667 - va
l_loss: 1.0941
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 99s 12s/step - accuracy: 0.5840 - loss: 1.0850 - val_accuracy: 0.3167 - val
_loss: 1.0194
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 103s 13s/step - accuracy: 0.4299 - loss: 1.0535 - val_accuracy: 0.3000 - va
l_loss: 3.0112
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 100s 12s/step - accuracy: 0.3782 - loss: 1.4493 - val_accuracy: 0.6167 - va
l_loss: 1.0051
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 100s 12s/step - accuracy: 0.5150 - loss: 0.9994 - val_accuracy: 0.6333 - va
l_loss: 0.7940
Epoch 10/30
3/8 ━━━━━━━━━━━━━━━━━━━━ 1:08 14s/step - accuracy: 0.6493 - loss: 0.7376
```

In [ ]:

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

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Train and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Train and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

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

In [ ]:

```
model.save('BestModel_VGG-16_H2O.h5')
```

In [ ]:

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

model = load_model(r'D:\Kuliah\SEM 5\ml\UAS\UASSSSS\BestModel_VGG-16_H2O.h5')
class_names = ['JamurShitake', 'JamurReishi', 'JamurKuping']

def classify_and_save_image(image_path, save_path='predicted_image.jpg'):
    try:
        img = image.load_img(image_path, target_size=(224, 224))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)

        predictions = model.predict(img_array)
        predicted_class = np.argmax(predictions[0])
        confidence = np.max(predictions[0]) * 100

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

        img.save(save_path)

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

result = classify_and_save_image(r'D:\Kuliah\SEM 5\ml\UAS\UASSSSS\datasetUASML\test_data\shitake\shitake (9).jpg'
, save_path='jamurShitake.jpg')
print(result)
```

In [ ]:

```
import tensorflow as tf
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    r'D:\Kuliah\SEM 5\ml\UAS\UASSSSS\datasetUASML\test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(224, 224),
    shuffle=False
)

class_names = ['JamurKuping', 'JamurReishi', 'JamurShitake']

y_true = []
y_pred = []

for images, labels in test_ds:
    predictions = model.predict(images)
    y_true.extend(np.argmax(labels.numpy(), axis=1))
    y_pred.extend(np.argmax(predictions, axis=1))

cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

accuracy = np.trace(cm) / np.sum(cm)

precision = np.diag(cm) / (np.sum(cm, axis=0) + 1e-6)
recall = np.diag(cm) / (np.sum(cm, axis=1) + 1e-6)
f1_score = 2 * (precision * recall) / (precision + recall + 1e-6)

print("Confusion Matrix:\n", cm)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
```

Beniditto Eka Viyantyo - 220711605

Kelompok H20

Klasifikasi Jamur

AxelNet

In [18]:

```
import tensorflow as tf
import cv2
import numpy as np
from matplotlib import pyplot as plt
#load data
data_dir = r"D:\Ditto\Kuliah\Matkul\Semester 5\ML\Pertemuan 16(UAS)\UAS\datatest\datasetUASML\train_data"
#Randomize data yang telah di load sekaligus resize menjadi 180 x 180
data = tf.keras.utils.image_dataset_from_directory(data_dir, seed = 123, image_size=(180, 180), batch_size=16)
print(data.class_names)

class_names = data.class_names
```

```
Found 300 files belonging to 3 classes.
['JamurKuping', 'JamurReishi', 'JamurShitake']
```

In [19]:

```
img_size = 180
batch = 32
validation_split = 0.1

dataset = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch
)
```

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

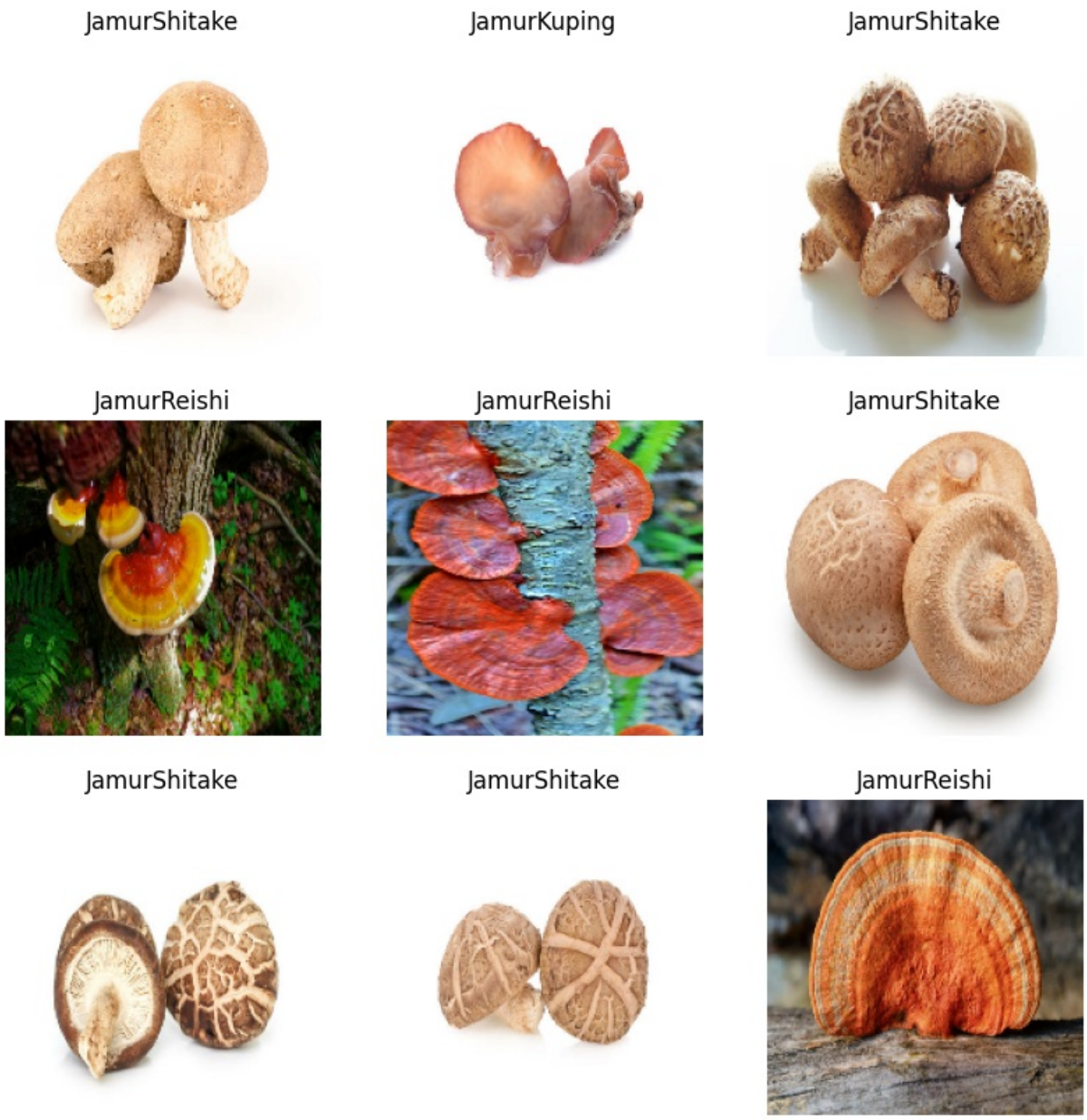
In [20]:

```
total_count = len(dataset)
val_count = int(total_count * validation_split)
train_count = total_count - val_count

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

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

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

In [21]:

```
import matplotlib.pyplot as plt

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

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

###Terdapat code yang hilang disini! lihat modul untuk menemukanya
###Pastikan nama kelas berhasil untuk ditampilkan
```



In [22]:

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

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

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

In [23]:

```
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

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

#Augmentasi data dengan menggunakan Sequential
data_augmentation = Sequential([
    layers.RandomFlip('horizontal', input_shape=(img_size, img_size, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])

plt.figure(figsize=(10, 10))
#Lihat data setelah di augmentasi
for images, labels in train_ds.take(1):
    images = data_augmentation(images)
    for i in range(9):
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.axis('off')
```

```
C:\Users\lenovo\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\preprocessing\tf_dat
a_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

In [24]:

```
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 alexnet(input_shape, n_classes):
    input_layer = Input(input_shape)
    x = Conv2D(96, (11, 11), strides=4, activation='relu')(input_layer)
    x = MaxPool2D(pool_size=(3, 3), strides=2)(x)

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

    x = Conv2D(384, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(384, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)
    x = MaxPool2D(pool_size=(3, 3), strides=2)(x)

    x = Flatten()(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(n_classes, activation='softmax')(x)

    model = Model(input_layer, output)
    return model

#Pastikan input shae dan jumlah kelas sesuai
input_shape = (img_size, img_size, 3)
n_classes = len(class_names)

#Clear Cache Keras menggunakan clear session
tf.keras.backend.clear_session()
model = alexnet(input_shape, n_classes)
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 43, 43, 96)	34,944
max_pooling2d (MaxPooling2D)	(None, 21, 21, 96)	0
conv2d_1 (Conv2D)	(None, 21, 21, 256)	614,656
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 256)	0
conv2d_2 (Conv2D)	(None, 10, 10, 384)	885,120
conv2d_3 (Conv2D)	(None, 10, 10, 384)	1,327,488
conv2d_4 (Conv2D)	(None, 10, 10, 256)	884,992
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 4096)	16,781,312
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16,781,312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 3)	12,291

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

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

Non-trainable params: 0 (0.00 B)

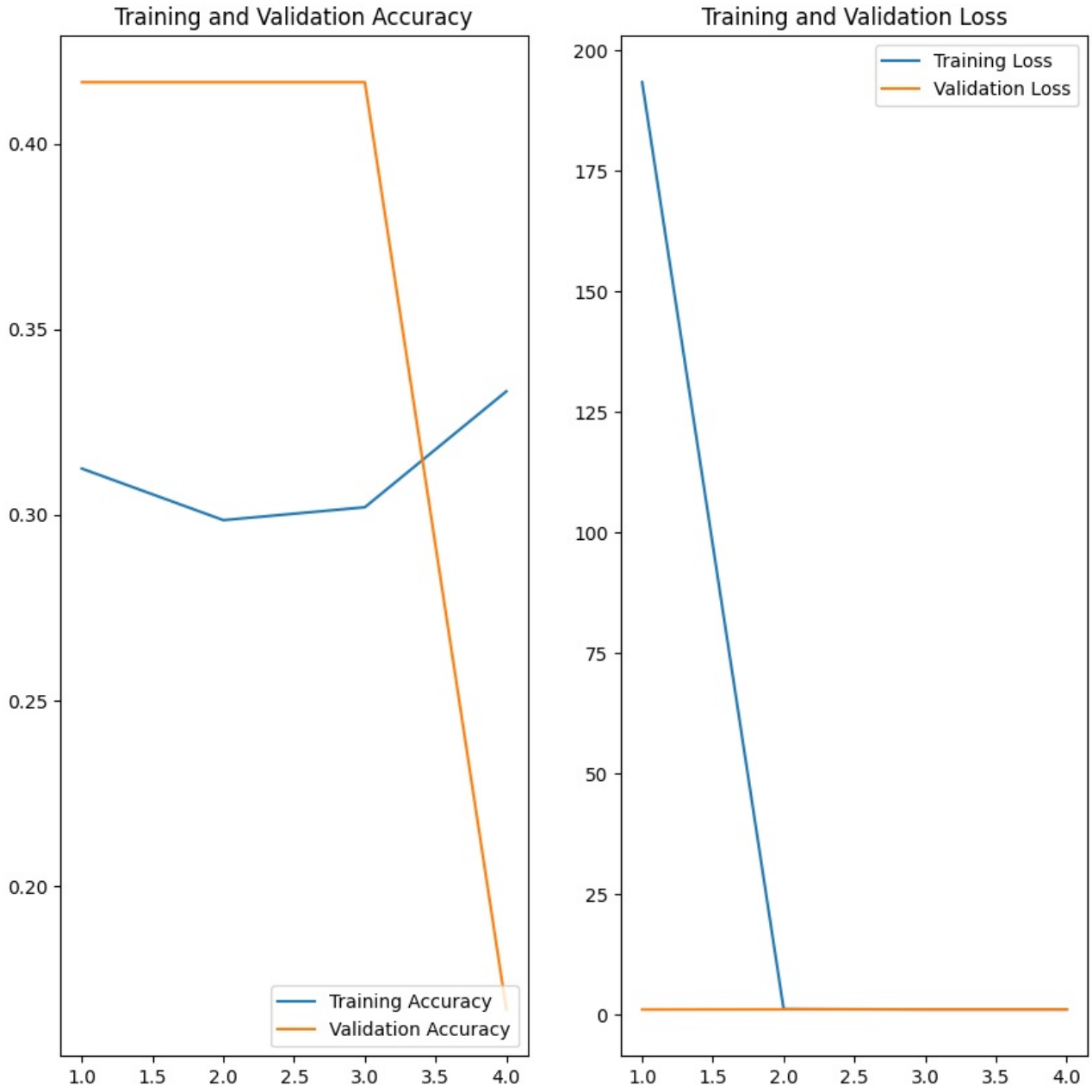
In [25]:

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
#Coimpile dengan optimizer adam
model.compile(
    optimizer=Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

#buat early stopping
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3, mode='max')

#fit validation data ke dalam model
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    callbacks=[early_stopping]
)
```

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 787ms/step - accuracy: 0.2912 - loss: 333.8502 - val_accuracy: 0.4167 -
val_loss: 1.0665
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 7s 723ms/step - accuracy: 0.3050 - loss: 1.1634 - val_accuracy: 0.4167 - va
l_loss: 1.1004
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 7s 745ms/step - accuracy: 0.3044 - loss: 1.0993 - val_accuracy: 0.4167 - va
l_loss: 1.1039
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 7s 746ms/step - accuracy: 0.3286 - loss: 1.0990 - val_accuracy: 0.1667 - va
l_loss: 1.1051
```

In [26]:

```
#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()
```



In [27]:

```
model.save('axelnet13.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_mode
l(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 [28]:

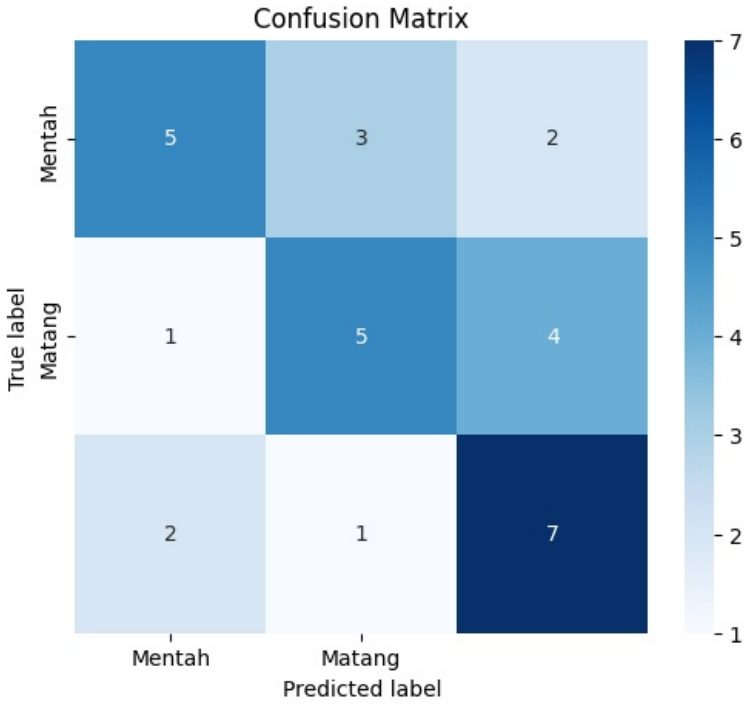
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

# Load the trained model
model = load_model(r'D:\Ditto\Kuliah\Matkul\Semester 5\ML\Pertemuan 16(UAS)\fixaxelnet.h5')  # Ganti dengan path
model Anda
class_names = ['JamurKuping', 'JamurReishi', 'JamurShitake']

# Function to classify images and save the original image
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        # Load and preprocess the image
        input_image = tf.keras.utils.load_img(image_path, target_size=(180, 180))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)  # Add batch dimension

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

        # Display prediction and confidence in notebook
        print(f"Prediksi: {class_names[class_idx]}")
        print(f"Confidence: {confidence:.2f}%")

        # Save the original image (without text)
        input_image = Image.open(image_path)
        input_image.save(save_path)

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

# Contoh penggunaan fungsi
result = classify_images(r'D:\Ditto\Kuliah\Matkul\Semester 5\ML\Pertemuan 16(UAS)\UAS\datatest\datasetUASML\test_
data\kuping\kuping (28).jpg', save_path='kuping.jpg')
print(result)
```

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

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 156ms/step
Prediksi: JamurKuping
Confidence: 49.82%
Prediksi: JamurKuping dengan confidence 49.82%. Gambar asli disimpan di kuping.jpg.
```

In [52]:

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

#memuat model yang telah dilatih sebelumnya
mobileNet_model = load_model(r'D:\Ditto\Kuliah\Matkul\Semester 5\ML\Pertemuan 16(UAS)\fixaxelnet.h5')#gunakan pat
h masing masing ya

#memuat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'D:\Ditto\Kuliah\Matkul\Semester 5\ML\Pertemuan 16(UAS)\UAS\datatest\datasetUASML\test_data', #direktori dat
a uji
    labels='inferred', #label otomatis dari subfolder yang ada
    label_mode='categorical',  #menghasilkan label dalam bentuk one-hot encoding
    batch_size=32, #ukuran batch untuk pemrosesan
    image_size=(180, 180) #ukuran gambar yang akan diproses
)

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

#ekstrak label sebenarnya dari test_data dan konversi ke bentuk indeks kelas
true_labels = [] #menyimpan label asli dalam bentuk indeks
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())  #konversi one-hot ke 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 set
iap sel matriks
                                                                 #fmt='d' untuk menampilkan bilangan bulat tanpa
desimal
            xticklabels=["Mentah", "Matang"], yticklabels=["Mentah", "Matang"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

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

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

Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 276ms/step
```




```
Confusion Matrix:
 [[5 3 2]
 [1 5 4]
 [2 1 7]]
Akurasi: 0.5666666666666667
Presisi: [0.625      0.55555556 0.53846154]
Recall: [0.5 0.5 0.7]
F1 Score: [0.55555556 0.52631579 0.60869565]
```

Aiko Evelyn 220711768 H2O Klasifikasi Jenis Jamur MobileNet

In [113]:

```
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 [114]:

```
count = 0
dirs = os.listdir(r'D:\Kuliah\SEMESTER 5\ML\UAS\train_data')
for dir in dirs:
    files = list(os.listdir(r'D:\Kuliah\SEMESTER 5\ML\UAS\train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
JamurKuping Folder has 100 Images
JamurReishi Folder has 100 Images
JamurShitake Folder has 100 Images
Images Folder has 300 Images
```

In [115]:

```
base_dir = r'D:\Kuliah\SEMESTER 5\ML\UAS\train_data'
img_size = 180
batch= 32
validation_split = 0.1
```

In [116]:

```
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 [117]:

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

```
Class Names: ['JamurKuping', 'JamurReishi', 'JamurShitake']
```

Train-Validation-Test Split

In [118]:

```
train_split = 0.8
validation_split = 0.1
test_split = 0.1

total_count = len(dataset)

train_count = int(total_count * train_split)
val_count = int(total_count * validation_split)
test_count = total_count - train_count - val_count

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

train_ds = dataset.take(train_count)
remaining_ds = dataset.skip(train_count)

val_ds = remaining_ds.take(val_count)
test_ds = remaining_ds.skip(val_count)
```

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

In [119]:

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

In [120]:

```
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 [121]:

```
import numpy as np

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

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

In [122]:

```
AUTOTUNE = tf.data.AUTOTUNE
```

In [123]:

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

In [124]:

```
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

Data Augmentation

In [125]:

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

```
C:\Users\pf34h\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\preprocessing\tf_data
_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

In [126]:

```
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 [127]:

```
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)
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')
])
```

```
C:\Users\pf34h\AppData\Local\Temp\ipykernel_3240\191715207.py:4: UserWarning: `input_shape` is undef
ined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224) wil
l be loaded as the default.
  base_model = MobileNet(include_top=False, input_shape=(img_size, img_size, 3))
```

In [128]:

```
from tensorflow.keras.optimizers import Adam

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

In [129]:

```
model.summary()
```

Model: "sequential_11"

Layer (type)	Output Shape	Param #
sequential_10 (Sequential)	(None, 180, 180, 3)	0
rescaling_5 (Rescaling)	(None, 180, 180, 3)	0
mobilenet_1.00_224 (Functional)	(None, 5, 5, 1024)	3,228,864
global_average_pooling2d_5 (GlobalAveragePooling2D)	(None, 1024)	0
dense_10 (Dense)	(None, 128)	131,200
dropout_5 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 3)	387

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

Trainable params: 131,587 (514.01 KB)

Non-trainable params: 3,228,864 (12.32 MB)

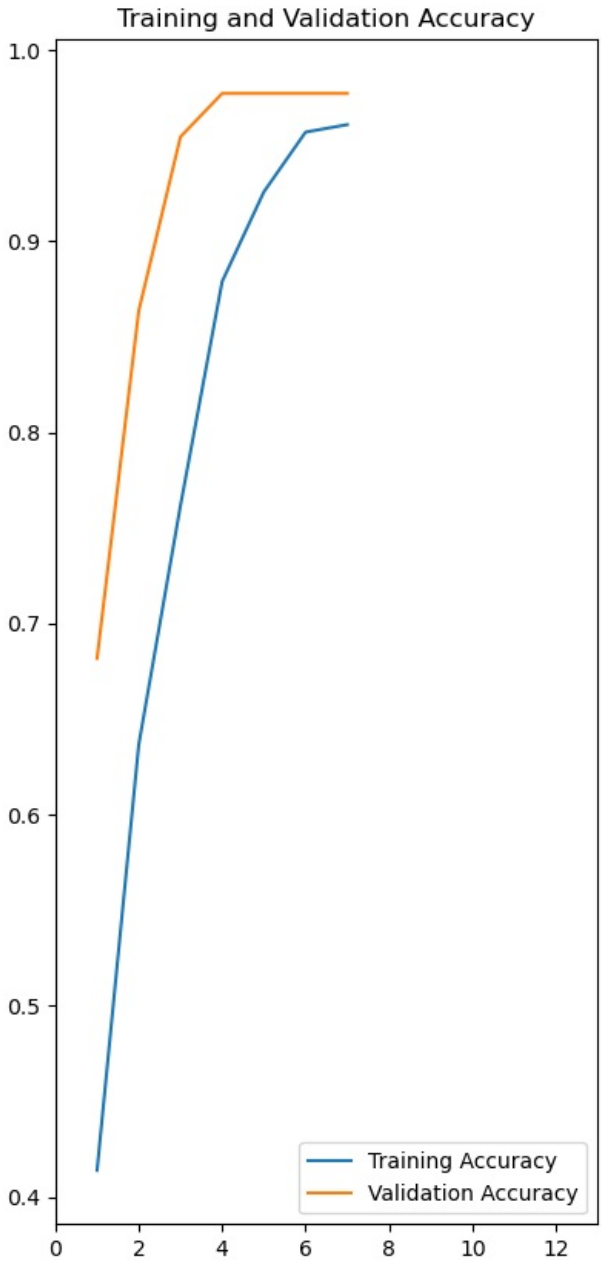
In [130]:

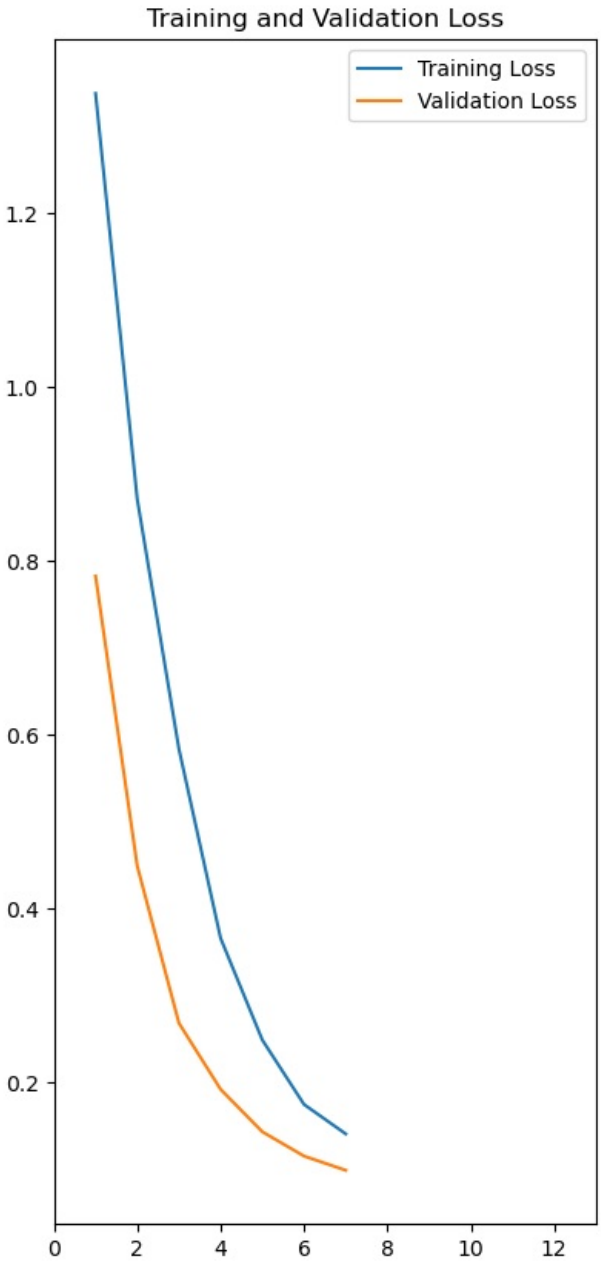
```
from tensorflow.keras.callbacks import EarlyStopping

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

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

```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 14s 619ms/step - accuracy: 0.3851 - loss: 1.4373 - val_accuracy: 0.6818 - v
al_loss: 0.7829
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 398ms/step - accuracy: 0.6156 - loss: 0.9067 - val_accuracy: 0.8636 - va
l_loss: 0.4493
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 409ms/step - accuracy: 0.7282 - loss: 0.6568 - val_accuracy: 0.9545 - va
l_loss: 0.2686
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 435ms/step - accuracy: 0.8793 - loss: 0.3787 - val_accuracy: 0.9773 - va
l_loss: 0.1924
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 409ms/step - accuracy: 0.9046 - loss: 0.2572 - val_accuracy: 0.9773 - va
l_loss: 0.1436
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 402ms/step - accuracy: 0.9515 - loss: 0.1725 - val_accuracy: 0.9773 - va
l_loss: 0.1157
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 392ms/step - accuracy: 0.9459 - loss: 0.1539 - val_accuracy: 0.9773 - va
l_loss: 0.0993
```

In [131]:

```
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 [132]:

```
model.save('BestModel_MobileNet_H2O.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_mode
l(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 [133]:

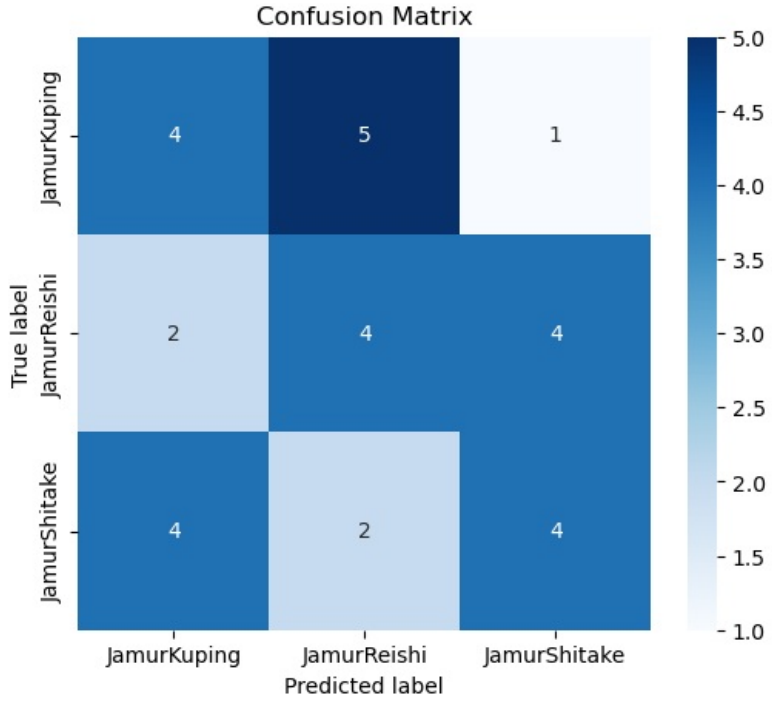
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

model = load_model(r'D:\Kuliah\SEMESTER 5\ML\UAS\BestModel_MobileNet_H2O.h5')
class_names = ['JamurKuping', 'JamurReishi', 'JamurShitake']

def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        input_image = tf.keras.utils.load_img(image_path, target_size=(180, 180))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)

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


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


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

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

result = classify_images(r'D:\Kuliah\SEMESTER 5\ML\UAS\test_data\reishi\reishi (9).jpg', save_path='reishi.jpg')
print(result)
```

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

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Prediksi: JamurReishi
Confidence: 57.32%
Prediksi: JamurReishi dengan confidence 57.32%. Gambar asli disimpan di reishi.jpg.
```

In [134]:

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

# Muat model
mobileNet_model = load_model(r'D:\Kuliah\SEMESTER 5\ML\UAS\BestModel_MobileNet_H2O.h5')

# Kompilasi ulang model
mobileNet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Load dataset untuk pengujian
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'D:\Kuliah\SEMESTER 5\ML\UAS\test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

# Prediksi menggunakan model
y_pred = mobileNet_model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)

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

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

# Menghitung metrik evaluasi
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)

# Plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["JamurKuping", "JamurReishi", "JamurShitake"],
            yticklabels=["JamurKuping", "JamurReishi", "JamurShitake"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

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

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

Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```




```
Confusion Matrix:
 [[4 5 1]
 [2 4 4]
 [4 2 4]]
Akurasi: 0.4
Presisi: [0.4        0.36363636 0.44444444]
Recall: [0.4 0.4 0.4]
F1 Score: [0.4        0.38095238 0.42105263]
```

Aiko Evelyn 220711768 H2O Klasifikasi Jenis Jamur GoogleNet

In [52]:

```
import tensorflow as tf

import numpy as np
from matplotlib import pyplot as plt

data_dir = r"D:\Kuliah\SEMESTER 5\ML\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
```

```
Found 300 files belonging to 3 classes.
['JamurKuping', 'JamurReishi', 'JamurShitake']
```

In [53]:

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

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

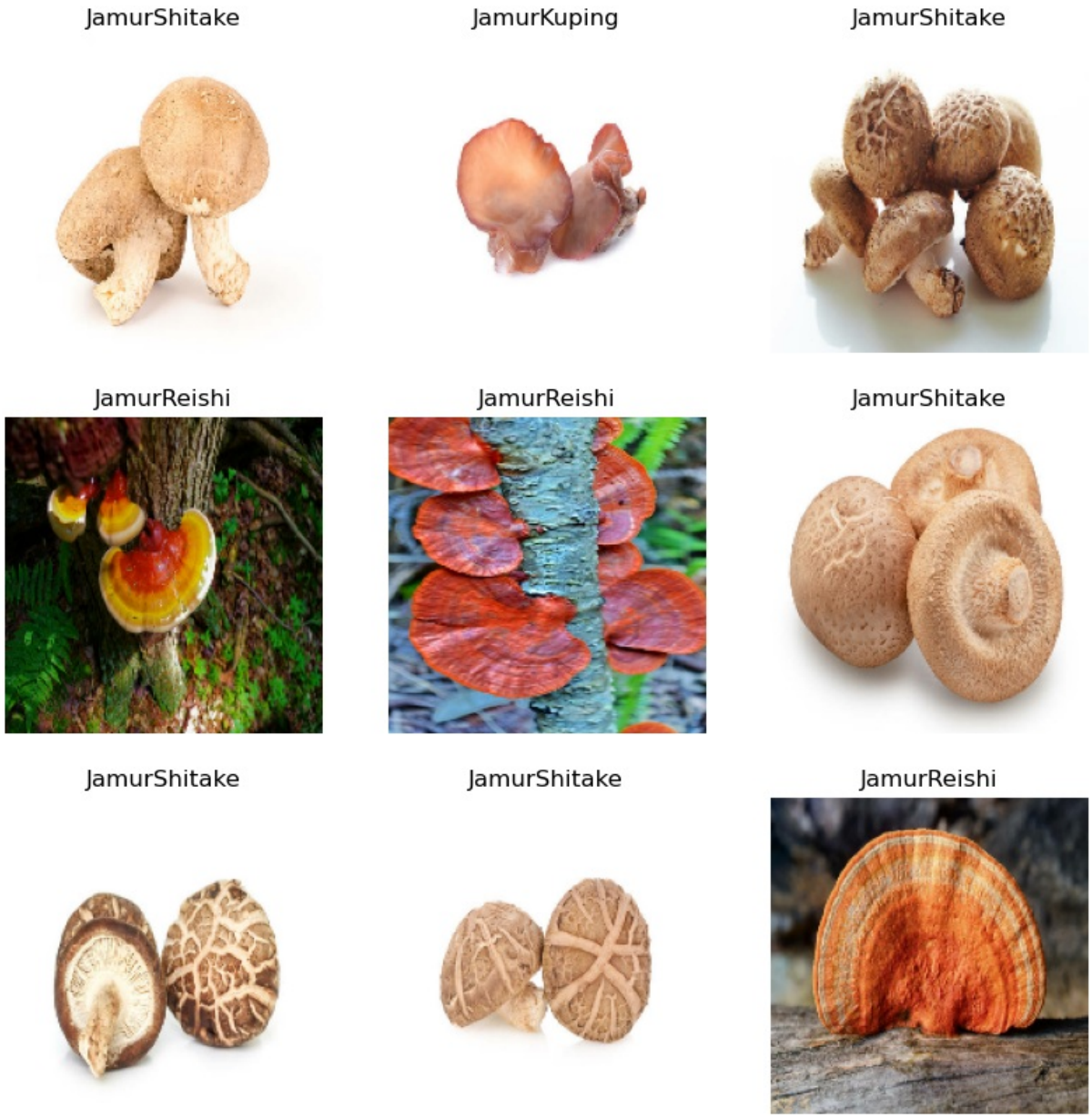
In [54]:

```
train_split = 0.8
validation_split = 0.1
test_split = 0.1

total_count = len(dataset)

train_count = int(total_count * train_split)
val_count = int(total_count * validation_split)
test_count = total_count - train_count - val_count

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

train_ds = dataset.take(train_count)
remaining_ds = dataset.skip(train_count)

val_ds = remaining_ds.take(val_count)
test_ds = remaining_ds.skip(val_count)
```

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

In [55]:

```
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 [56]:

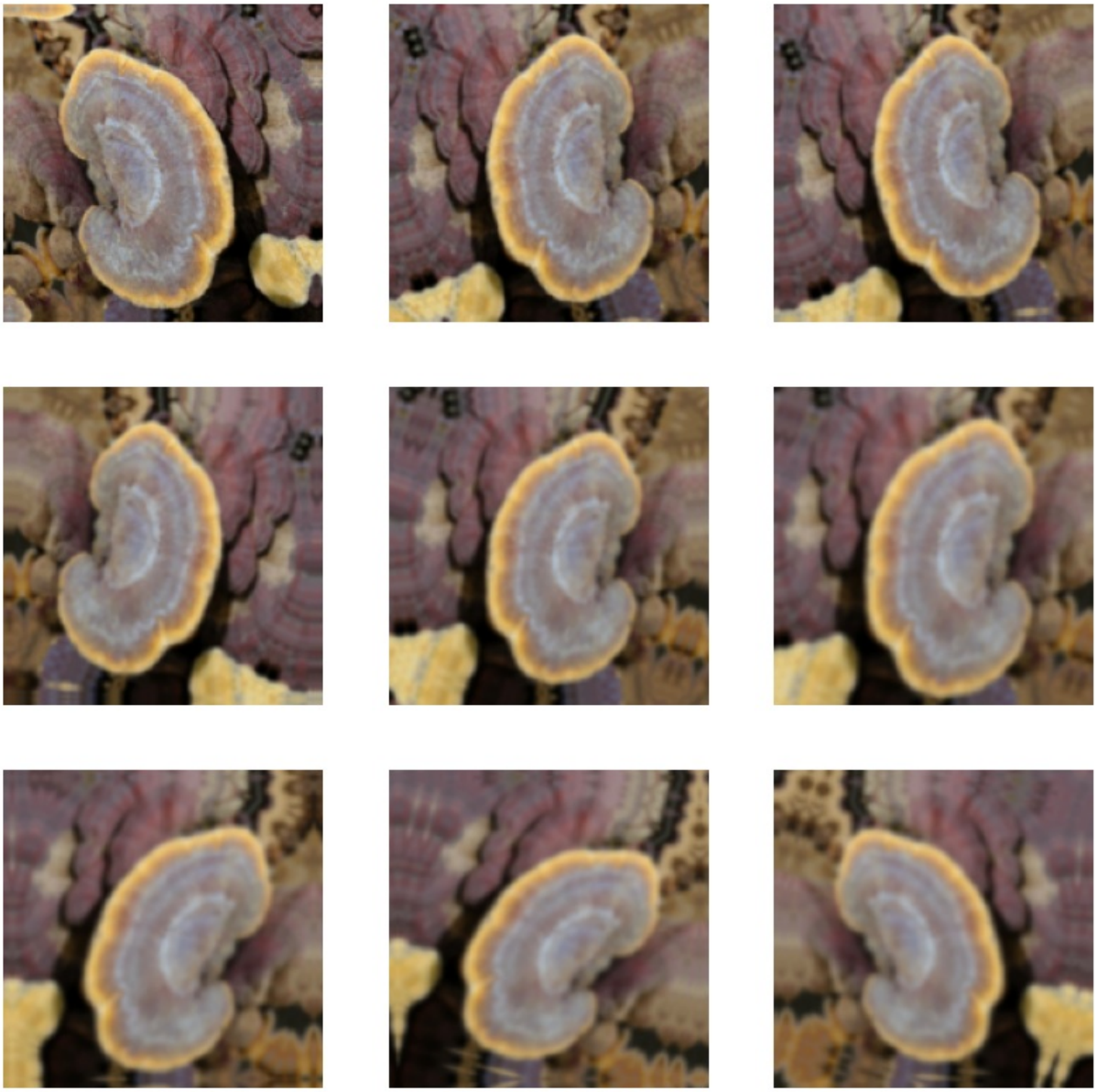
```
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

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

In [57]:

```
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:\Users\pf34h\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\preprocessing\tf_data
_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```



In [58]:

```
import tensorflow as tf
import keras

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

from keras._tf_keras.keras.models import load_model

def googlenet(input_shape, n_classes):

    def inception_block(x, f):
        t1 = Conv2D(f[0], 1, activation='relu')(x)

        t2 = Conv2D(f[1], 1, activation='relu')(x)
        t2 = Conv2D(f[2], 3, padding='same', activation='relu')(t2)

        t3 = Conv2D(f[3], 1, activation='relu')(x)
        t3 = Conv2D(f[4], 5, padding='same', activation='relu')(t3)

        t4 = MaxPool2D(3, 1, padding='same')(x)
        t4 = Conv2D(f[5], 1, activation='relu')(t4)

        output = Concatenate()([t1, t2, t3, t4])
        return output

    input = Input(input_shape)

    x = Conv2D(64, 7, strides=2, padding='same', activation='relu')(input)
    x = MaxPool2D(3, strides=2, padding='same')(x)

    x = Conv2D(64, 1, activation='relu')(x)
    x = Conv2D(192, 3, padding='same', activation='relu')(x)
    x = MaxPool2D(3, strides=2)(x)

    x = inception_block(x, [64, 96, 128, 16, 32, 32])
    x = inception_block(x, [128, 128, 192, 32, 96, 64])
    x = MaxPool2D(3, strides=2, padding='same')(x)

    x = inception_block(x, [192, 96, 208, 16, 48, 64])
    x = inception_block(x, [160, 112, 224, 24, 64, 64])
    x = inception_block(x, [128, 128, 256, 24, 64, 64])
    x = inception_block(x, [112, 144, 288, 32, 64, 64])
    x = inception_block(x, [256, 160, 320, 32, 128, 128])
    x = MaxPool2D(3, strides=2, padding='same')(x)

    x = inception_block(x, [256, 160, 320, 32, 128, 128])
    x = inception_block(x, [384, 192, 384, 48, 128, 128])

    x = AvgPool2D(3, strides=1)(x)
    x = Dropout(0.4)(x)

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

    model = Model(input, output)
    return model

input_shape = 180, 180, 3
n_classes = 3


K.clear_session()

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

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 180, 180, 3)	0	-
conv2d (Conv2D)	(None, 90, 90, 64)	9,472	input_layer[0][0]
max_pooling2d	(None, 45, 45,	0	conv2d[0][0]

(MaxPooling2D)	64)		
conv2d_1 (Conv2D)	(None, 45, 45, 64)	4,160	max_pooling2d[0]…
conv2d_2 (Conv2D)	(None, 45, 45, 192)	110,784	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 22, 22, 192)	0	conv2d_2[0][0]
conv2d_4 (Conv2D)	(None, 22, 22, 96)	18,528	max_pooling2d_1[…
conv2d_6 (Conv2D)	(None, 22, 22, 16)	3,088	max_pooling2d_1[…
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 192)	0	max_pooling2d_1[…
conv2d_3 (Conv2D)	(None, 22, 22, 64)	12,352	max_pooling2d_1[…
conv2d_5 (Conv2D)	(None, 22, 22, 128)	110,720	conv2d_4[0][0]
conv2d_7 (Conv2D)	(None, 22, 22, 32)	12,832	conv2d_6[0][0]
conv2d_8 (Conv2D)	(None, 22, 22, 32)	6,176	max_pooling2d_2[…
concatenate (Concatenate)	(None, 22, 22, 256)	0	conv2d_3[0][0], conv2d_5[0][0], conv2d_7[0][0], conv2d_8[0][0]
conv2d_10 (Conv2D)	(None, 22, 22, 128)	32,896	concatenate[0][0]
conv2d_12 (Conv2D)	(None, 22, 22, 32)	8,224	concatenate[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 22, 22, 256)	0	concatenate[0][0]
conv2d_9 (Conv2D)	(None, 22, 22, 128)	32,896	concatenate[0][0]
conv2d_11 (Conv2D)	(None, 22, 22, 192)	221,376	conv2d_10[0][0]
conv2d_13 (Conv2D)	(None, 22, 22, 96)	76,896	conv2d_12[0][0]
conv2d_14 (Conv2D)	(None, 22, 22, 64)	16,448	max_pooling2d_3[…
concatenate_1 (Concatenate)	(None, 22, 22, 480)	0	conv2d_9[0][0], conv2d_11[0][0], conv2d_13[0][0], conv2d_14[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 11, 11, 480)	0	concatenate_1[0]…
conv2d_16 (Conv2D)	(None, 11, 11, 96)	46,176	max_pooling2d_4[…
conv2d_18 (Conv2D)	(None, 11, 11, 16)	7,696	max_pooling2d_4[…
max_pooling2d_5 (MaxPooling2D)	(None, 11, 11, 480)	0	max_pooling2d_4[…
conv2d_15 (Conv2D)	(None, 11, 11, 192)	92,352	max_pooling2d_4[…
conv2d_17 (Conv2D)	(None, 11, 11, 208)	179,920	conv2d_16[0][0]
conv2d_19 (Conv2D)	(None, 11, 11, 48)	19,248	conv2d_18[0][0]

conv2d_20 (Conv2D)	(None, 11, 11, 64)	30,784	max_pooling2d_5[…
concatenate_2 (Concatenate)	(None, 11, 11, 512)	0	conv2d_15[0][0], conv2d_17[0][0], conv2d_19[0][0], conv2d_20[0][0]
conv2d_22 (Conv2D)	(None, 11, 11, 112)	57,456	concatenate_2[0]…
conv2d_24 (Conv2D)	(None, 11, 11, 24)	12,312	concatenate_2[0]…
max_pooling2d_6 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_2[0]…
conv2d_21 (Conv2D)	(None, 11, 11, 160)	82,080	concatenate_2[0]…
conv2d_23 (Conv2D)	(None, 11, 11, 224)	226,016	conv2d_22[0][0]
conv2d_25 (Conv2D)	(None, 11, 11, 64)	38,464	conv2d_24[0][0]
conv2d_26 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_6[…
concatenate_3 (Concatenate)	(None, 11, 11, 512)	0	conv2d_21[0][0], conv2d_23[0][0], conv2d_25[0][0], conv2d_26[0][0]
conv2d_28 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]…
conv2d_30 (Conv2D)	(None, 11, 11, 24)	12,312	concatenate_3[0]…
max_pooling2d_7 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_3[0]…
conv2d_27 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]…
conv2d_29 (Conv2D)	(None, 11, 11, 256)	295,168	conv2d_28[0][0]
conv2d_31 (Conv2D)	(None, 11, 11, 64)	38,464	conv2d_30[0][0]
conv2d_32 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_7[…
concatenate_4 (Concatenate)	(None, 11, 11, 512)	0	conv2d_27[0][0], conv2d_29[0][0], conv2d_31[0][0], conv2d_32[0][0]
conv2d_34 (Conv2D)	(None, 11, 11, 144)	73,872	concatenate_4[0]…
conv2d_36 (Conv2D)	(None, 11, 11, 32)	16,416	concatenate_4[0]…
max_pooling2d_8 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_4[0]…
conv2d_33 (Conv2D)	(None, 11, 11, 112)	57,456	concatenate_4[0]…
conv2d_35 (Conv2D)	(None, 11, 11, 288)	373,536	conv2d_34[0][0]
conv2d_37 (Conv2D)	(None, 11, 11, 64)	51,264	conv2d_36[0][0]
conv2d_38 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_8[…
concatenate_5	(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, 160)	84,640	concatenate_5[0]…
conv2d_42 (Conv2D)	(None, 11, 11, 32)	16,928	concatenate_5[0]…
max_pooling2d_9 (MaxPooling2D)	(None, 11, 11, 528)	0	concatenate_5[0]…
conv2d_39 (Conv2D)	(None, 11, 11, 256)	135,424	concatenate_5[0]…
conv2d_41 (Conv2D)	(None, 11, 11, 320)	461,120	conv2d_40[0][0]
conv2d_43 (Conv2D)	(None, 11, 11, 128)	102,528	conv2d_42[0][0]
conv2d_44 (Conv2D)	(None, 11, 11, 128)	67,712	max_pooling2d_9[…
concatenate_6 (Concatenate)	(None, 11, 11, 832)	0	conv2d_39[0][0], conv2d_41[0][0], conv2d_43[0][0], conv2d_44[0][0]
max_pooling2d_10 (MaxPooling2D)	(None, 6, 6, 832)	0	concatenate_6[0]…
conv2d_46 (Conv2D)	(None, 6, 6, 160)	133,280	max_pooling2d_10…
conv2d_48 (Conv2D)	(None, 6, 6, 32)	26,656	max_pooling2d_10…
max_pooling2d_11 (MaxPooling2D)	(None, 6, 6, 832)	0	max_pooling2d_10…
conv2d_45 (Conv2D)	(None, 6, 6, 256)	213,248	max_pooling2d_10…
conv2d_47 (Conv2D)	(None, 6, 6, 320)	461,120	conv2d_46[0][0]
conv2d_49 (Conv2D)	(None, 6, 6, 128)	102,528	conv2d_48[0][0]
conv2d_50 (Conv2D)	(None, 6, 6, 128)	106,624	max_pooling2d_11…
concatenate_7 (Concatenate)	(None, 6, 6, 832)	0	conv2d_45[0][0], conv2d_47[0][0], conv2d_49[0][0], conv2d_50[0][0]
conv2d_52 (Conv2D)	(None, 6, 6, 192)	159,936	concatenate_7[0]…
conv2d_54 (Conv2D)	(None, 6, 6, 48)	39,984	concatenate_7[0]…
max_pooling2d_12 (MaxPooling2D)	(None, 6, 6, 832)	0	concatenate_7[0]…
conv2d_51 (Conv2D)	(None, 6, 6, 384)	319,872	concatenate_7[0]…
conv2d_53 (Conv2D)	(None, 6, 6, 384)	663,936	conv2d_52[0][0]
conv2d_55 (Conv2D)	(None, 6, 6, 128)	153,728	conv2d_54[0][0]
conv2d_56 (Conv2D)	(None, 6, 6, 128)	106,624	max_pooling2d_12…
concatenate_8 (Concatenate)	(None, 6, 6, 1024)	0	conv2d_51[0][0], conv2d_53[0][0], conv2d_55[0][0], conv2d_56[0][0]
average_pooling2d (AveragePooling2D)	(None, 4, 4, 1024)	0	concatenate_8[0]…
dropout (Dropout)	(None, 4, 4, 1024)	0	average_pooling2…
flatten (Flatten)	(None, 16384)	0	dropout[0][0]
dense (Dense)	(None, 3)	49,155	flatten[0][0]

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

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

Non-trainable params: 0 (0.00 B)

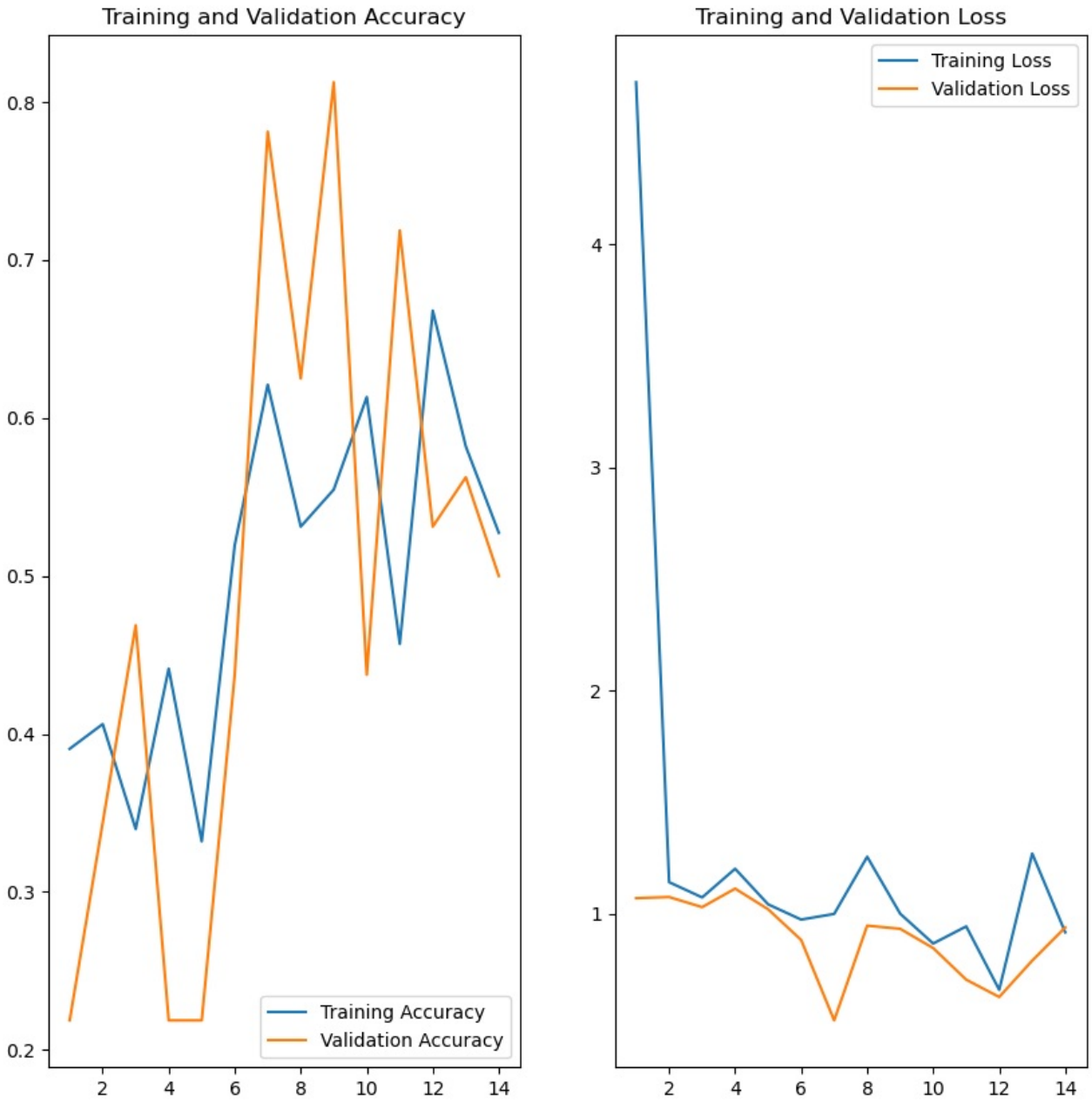
In [59]:

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

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


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

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

```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 28s 1s/step - accuracy: 0.4023 - loss: 7.0698 - val_accuracy: 0.2188 - val_
loss: 1.0698
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 882ms/step - accuracy: 0.4301 - loss: 1.1169 - val_accuracy: 0.3438 - va
l_loss: 1.0748
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 8s 1s/step - accuracy: 0.3493 - loss: 1.0910 - val_accuracy: 0.4688 - val_l
oss: 1.0295
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 875ms/step - accuracy: 0.4863 - loss: 1.1087 - val_accuracy: 0.2188 - va
l_loss: 1.1126
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 928ms/step - accuracy: 0.3285 - loss: 1.0650 - val_accuracy: 0.2188 - va
l_loss: 1.0199
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.4776 - loss: 0.9758 - val_accuracy: 0.4375 - val_l
oss: 0.8823
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.6067 - loss: 0.8442 - val_accuracy: 0.7812 - val_l
oss: 0.5220
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5553 - loss: 1.3774 - val_accuracy: 0.6250 - val_
loss: 0.9466
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 920ms/step - accuracy: 0.5242 - loss: 1.0101 - val_accuracy: 0.8125 - va
l_loss: 0.9324
Epoch 10/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 897ms/step - accuracy: 0.6320 - loss: 0.8997 - val_accuracy: 0.4375 - va
l_loss: 0.8460
Epoch 11/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 893ms/step - accuracy: 0.4384 - loss: 0.9741 - val_accuracy: 0.7188 - va
l_loss: 0.7044
Epoch 12/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 853ms/step - accuracy: 0.6686 - loss: 0.7020 - val_accuracy: 0.5312 - va
l_loss: 0.6267
Epoch 13/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 863ms/step - accuracy: 0.6627 - loss: 0.7394 - val_accuracy: 0.5625 - va
l_loss: 0.7899
Epoch 14/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 7s 952ms/step - accuracy: 0.5271 - loss: 0.8846 - val_accuracy: 0.5000 - va
l_loss: 0.9384
```

In [60]:

```
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()
```



In [61]:

```
model.save('BestModel_GoogleNet_H2O.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_mode
l(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 [62]:

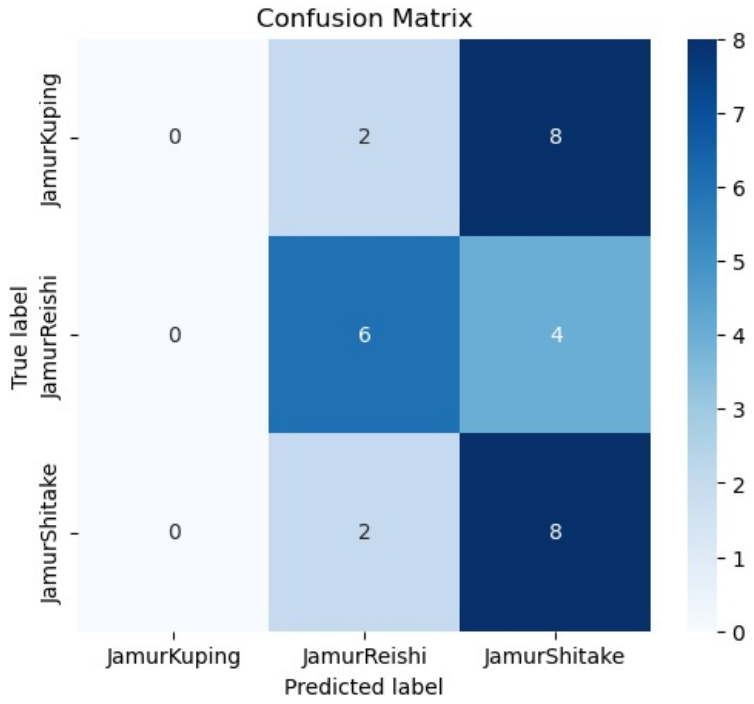
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image


model = load_model(r'D:\Kuliah\SEMESTER 5\ML\UAS\BestModel_GoogleNet_H2O.h5')
class_names = ['JamurKuping', 'JamurReishi', 'JamurShitake']


def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        input_image = tf.keras.utils.load_img(image_path, target_size=(180, 180))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)


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


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


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

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


result = classify_images(r'D:\Kuliah\SEMESTER 5\ML\UAS\test_data\kuping\kuping (6).jpg', save_path='kuping.jpg')
print(result)
```

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

1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Prediksi: JamurShitake
Confidence: 38.90%
Prediksi: JamurShitake dengan confidence 38.90%. Gambar asli disimpan di kuping.jpg.
```

In [63]:

```
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'D:\Kuliah\SEMESTER 5\ML\UAS\test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)


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


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


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


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


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


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


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


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

```
Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```




```
Confusion Matrix:
 [[0 2 8]
 [0 6 4]
 [0 2 8]]
Akurasi: 0.4666666666666667
Presisi: [nan 0.6 0.4]
Recall: [0.  0.6 0.8]
F1 Score: [       nan 0.6        0.53333333]
```