SQLAlchemy Anggota :

1. **Antonius Rosarianto Wisnu Putro/200710813**
2. **Ignatius Joti Argapoda Panggabean/200710862**
3. **Sebastian Willys Lambang/200710639**
4. **Jhonatan Emanuel Wangge/210711843**
5. **Yoga Jesay Tarigan/180709976**

Indentitas

Yoga Jesay Tarigan/180709976

SQLAlchemy_KlasifikasiJamurEnokiKancingKuping_AlexNet

Double-click (or enter) to edit

```
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

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
```

```
count = 0
dirs = os.listdir(r'train_data')
for dir in dirs:
    files = list(os.listdir(r'train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
JamurEnoki Folder has 100 Images
JamurKancing Folder has 100 Images
JamurKuping Folder has 100 Images
Images Folder has 300 Images
```

```
base_dir = r'train_data'
img_size = 180
batch = 32
validation_split = 0.1
test_split = 0.1

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

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

```
Found 300 files belonging to 3 classes.
Class Names: ['JamurEnoki', 'JamurKancing', 'JamurKuping']
```

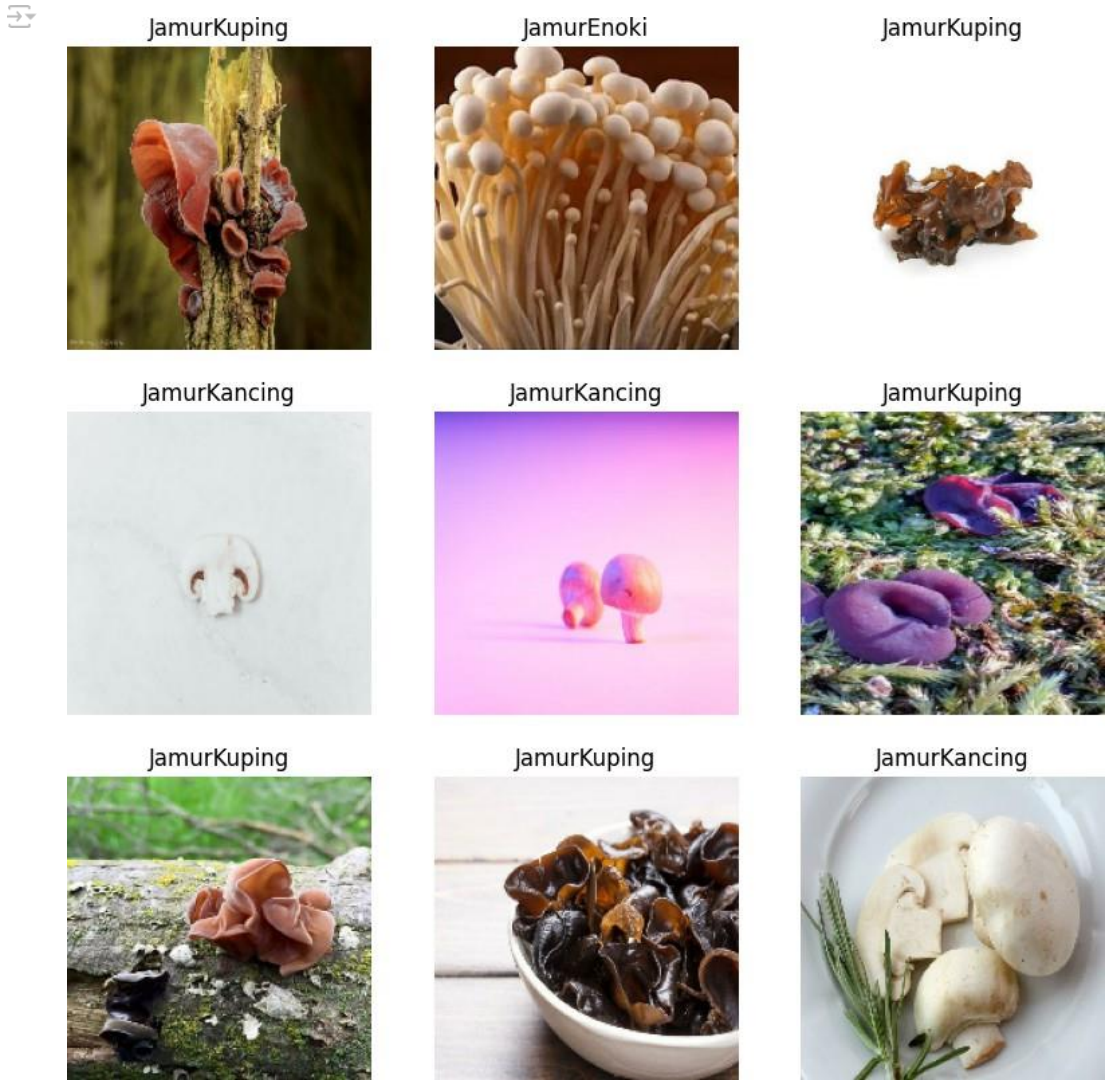
```
total_count = len(dataset)
test_count = int(total_count * test_split)
val_count = int(total_count * validation_split)
train_count = total_count - val_count - test_count

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

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

```
Test Images: 1
```

```
train_ds = dataset.take(train_count)
temp_ds = dataset.skip(train_count)
val_ds = temp_ds.take(val_count)
test_ds = temp_ds.skip(val_count)
```

```
import matplotlib.pyplot as plt

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

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

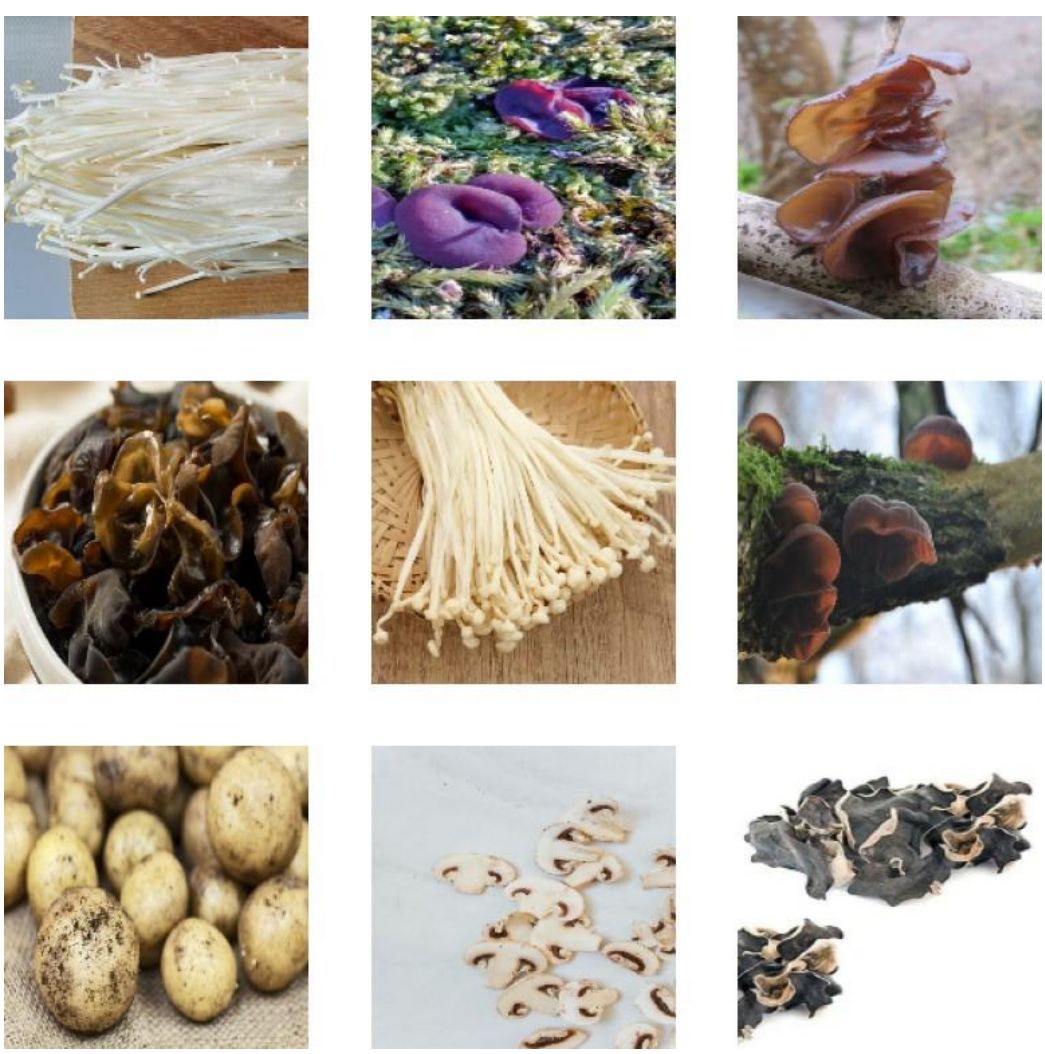



```
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

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

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


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"),
```

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

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

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

plt.show()
```



```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras import backend as K

def alexnet(input_shape, n_classes):
    model = Sequential()

    model.add(Conv2D(96, (11, 11), strides=4, activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(3, 3), strides=2))

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

    model.add(Conv2D(384, (3, 3), activation='relu', padding='same'))
    model.add(Conv2D(384, (3, 3), activation='relu', padding='same'))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(3, 3), strides=2))

    model.add(Flatten())
    model.add(Dense(4096, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(4096, activation='relu'))
```

```
    model.add(Dropout(0.5))
    model.add(Dense(n_classes, activation='softmax'))

    return model

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

K.clear_session()

model = alexnet(input_shape, n_classes)

model.summary()
```

```
WARNING:tensorflow:From c:\Python39\lib\site-packages\keras\src\backend\common\global_state.py:82: The name tf.reset_default_graph

c:\Python39\lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
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

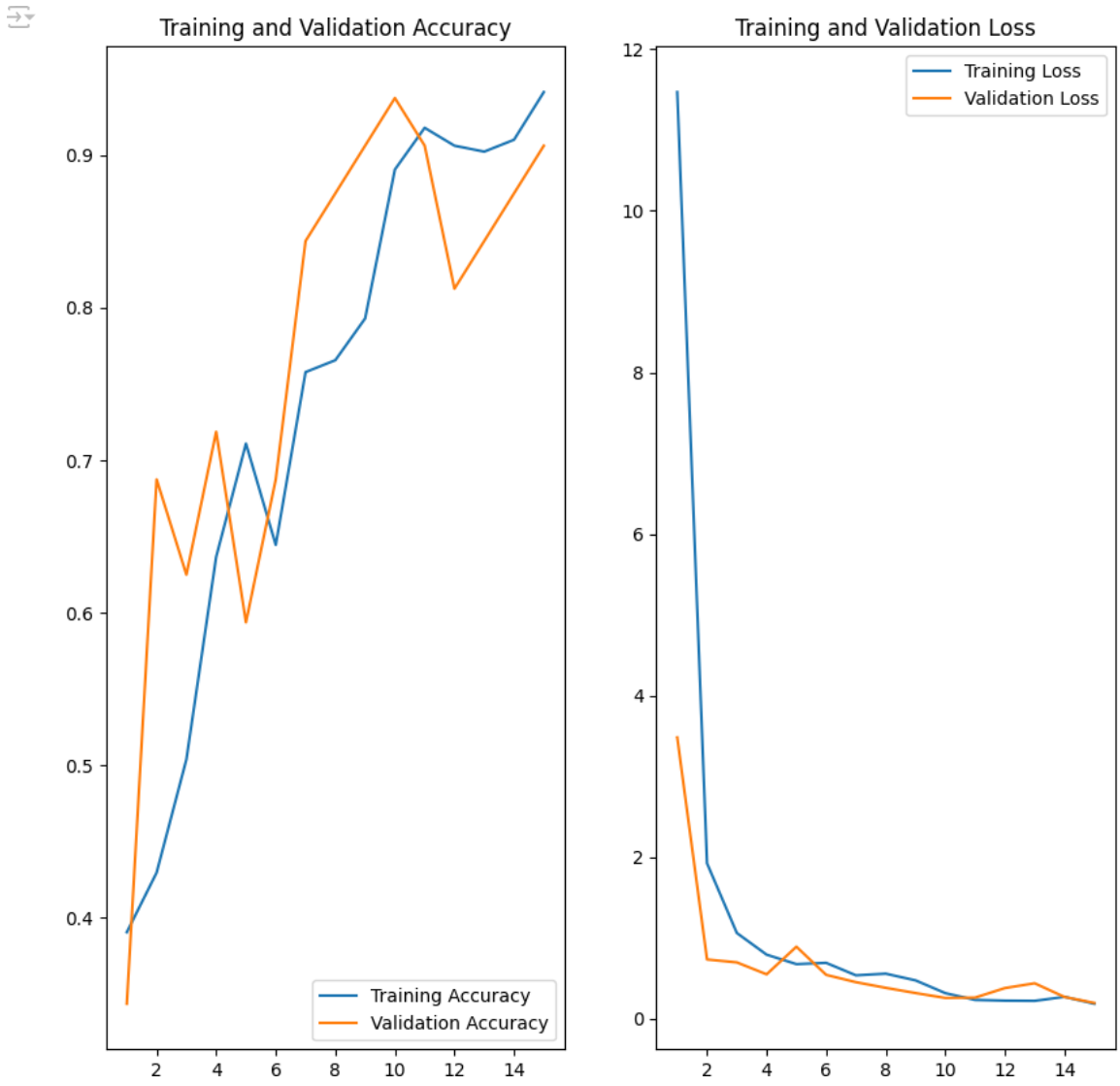
```
 Total params: 37,322,115 (142.37 MB)
 Trainable params: 37,322,115 (142.37 MB)
```

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

model.compile(
    optimizer=Adam(learning_rate=1e-4),
    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 ━━━━━━━━━━━━━━━━━━━━ 20s 2s/step - accuracy: 0.3502 - loss: 16.1997 - val_accuracy: 0.3438 - val_loss: 3.4821
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.4310 - loss: 2.1836 - val_accuracy: 0.6875 - val_loss: 0.7341
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.4880 - loss: 1.1130 - val_accuracy: 0.6250 - val_loss: 0.6980
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.6018 - loss: 0.8029 - val_accuracy: 0.7188 - val_loss: 0.5503
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 12s 1s/step - accuracy: 0.7150 - loss: 0.6700 - val_accuracy: 0.5938 - val_loss: 0.8920
Epoch 6/30
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.6539 - loss: 0.7010 - val_accuracy: 0.6875 - val_loss: 0.5435
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 12s 2s/step - accuracy: 0.7468 - loss: 0.5734 - val_accuracy: 0.8438 - val_loss: 0.4527
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step - accuracy: 0.7882 - loss: 0.5482 - val_accuracy: 0.8750 - val_loss: 0.3833
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 12s 2s/step - accuracy: 0.7852 - loss: 0.5033 - val_accuracy: 0.9062 - val_loss: 0.3186
Epoch 10/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.8789 - loss: 0.3344 - val_accuracy: 0.9375 - val_loss: 0.2578
Epoch 11/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 12s 2s/step - accuracy: 0.9260 - loss: 0.2359 - val_accuracy: 0.9062 - val_loss: 0.2623
Epoch 12/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.9119 - loss: 0.2174 - val_accuracy: 0.8125 - val_loss: 0.3801
Epoch 13/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 12s 1s/step - accuracy: 0.9020 - loss: 0.2322 - val_accuracy: 0.8438 - val_loss: 0.4394
Epoch 14/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.8918 - loss: 0.2856 - val_accuracy: 0.8750 - val_loss: 0.2703
Epoch 15/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 11s 1s/step - accuracy: 0.9465 - loss: 0.1632 - val_accuracy: 0.9062 - val_loss: 0.1965
```

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

plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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




```
model.save('BestModel_AlexNet_SQLAlchemy.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
```

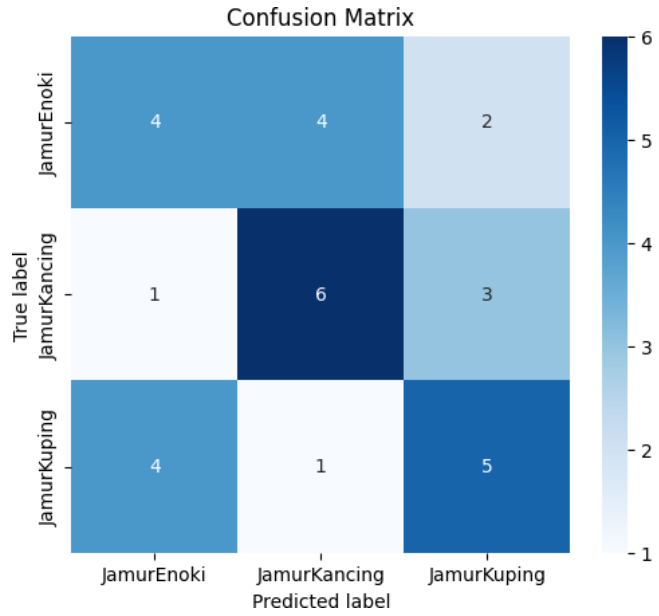
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

model = load_model(r'BestModel_AlexNet_SQLAlchemy.h5')
class_names = ['JamurEnoki', 'JamurKancing', 'JamurKuping']

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

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

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

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

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

result = classify_images(r'test_data\JamurKancing\JamurKancingTest_07.jpg', save_path='JamurKancingTest.jpg')
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 369ms/step
Prediksi: JamurKancing
Confidence: 39.39%
Prediksi: JamurKancing dengan confidence 39.39%. Gambar asli disimpan di JamurKancingTest.jpg.
```

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

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

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

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

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

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

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

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

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["JamurEnoki", "JamurKancing", "JamurKuping"], yticklabels=["JamurEnoki", "JamurKancing", "JamurKuping"])
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:
 [[4 4 2]
 [1 6 3]
 [4 1 5]]
Akurasi: 0.5
Presisi: [0.44444444 0.54545455 0.5       ]
Recall: [0.4 0.6 0.5]
F1 Score: [0.42105263 0.57142857 0.5       ]
```

Indentitas

Ignatius Joti Argapoda Panggabean/200710862

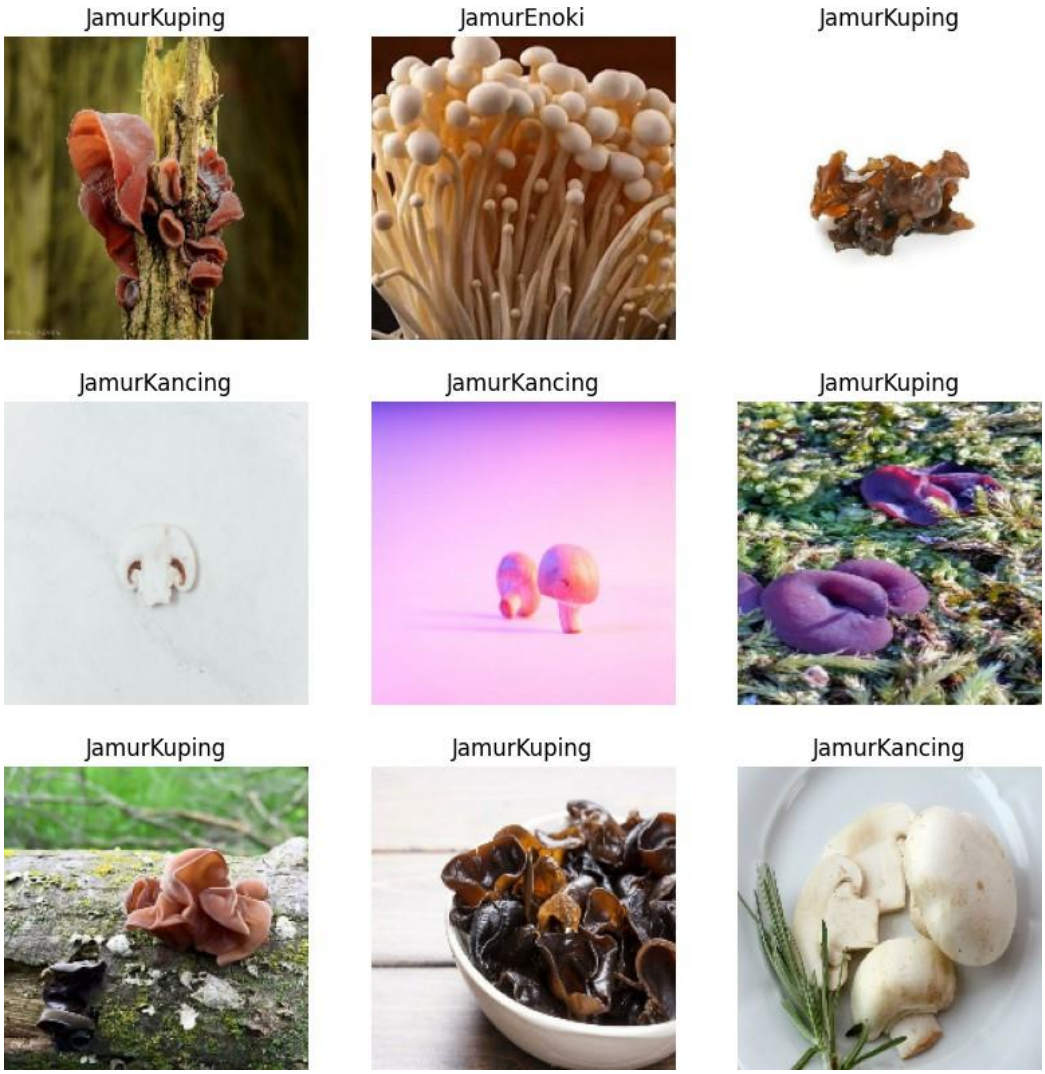
SQLAlchemy_KlasifikasiJamurEnokiKancingKuping_VGG-16

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

data_dir = r"train_data"

data = tf.keras.utils.image_dataset_from_directory(data_dir, seed = 123, image_size = (180, 180), batch_size = 16)
print(data.class_names)

class_names = data.class_names

img_size = 180
batch = 32
validation_split = 0.1
test_split = 0.1
dataset = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed = 123,
    image_size = (img_size, img_size),
    batch_size = batch,
)
total_count = len(dataset)
test_count = int(total_count * test_split)
val_count = int(total_count * validation_split)
train_count = total_count - val_count - test_count

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

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

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

```
Found 300 files belonging to 3 classes.
['JamurEnoki', 'JamurKancing', 'JamurKuping']
Found 300 files belonging to 3 classes.
Total Images: 10
Train Images: 8
Validation Images: 1
Test Images: 1
```

```
import matplotlib.pyplot as plt

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


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




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

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

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras import backend as K

def vgg16(input_shape, n_classes):
    model = Sequential()
```

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

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

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

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dense(n_classes, activation='softmax'))

    return model

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

K.clear_session()

model = vgg16(input_shape, n_classes)

model.summary()
```

```
c:\Python39\lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

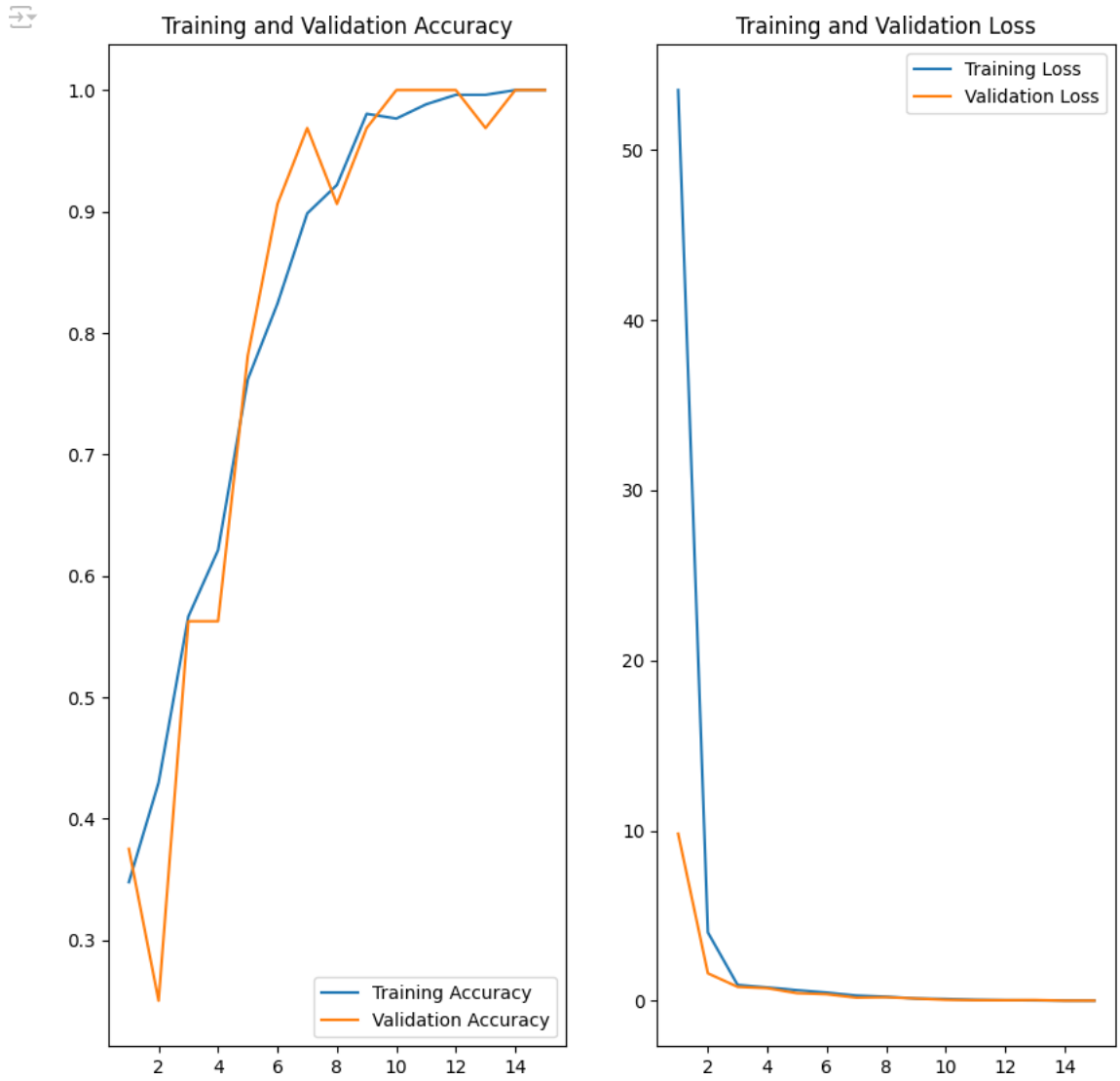
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 180, 180, 64)	1,792
conv2d_1 (Conv2D)	(None, 180, 180, 64)	36,928
max_pooling2d (MaxPooling2D)	(None, 90, 90, 64)	0
conv2d_2 (Conv2D)	(None, 90, 90, 128)	73,856
conv2d_3 (Conv2D)	(None, 90, 90, 128)	147,584
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 128)	0
conv2d_4 (Conv2D)	(None, 45, 45, 256)	295,168
conv2d_5 (Conv2D)	(None, 45, 45, 256)	590,080
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 256)	0
flatten (Flatten)	(None, 123904)	0
dense (Dense)	(None, 512)	63,439,360
dense_1 (Dense)	(None, 3)	1,539

Total params: 64,586,307 (246.38 MB)
Trainable params: 64,586,307 (246.38 MB)

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

model.compile(
    optimizer=Adam(learning_rate=1e-4),
    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 ━━━━━━━━━━━━━━━━━━━━ 128s 14s/step - accuracy: 0.3453 - loss: 58.7735 - val_accuracy: 0.3750 - val_loss: 9.8230
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 105s 13s/step - accuracy: 0.3667 - loss: 6.8227 - val_accuracy: 0.2500 - val_loss: 1.6225
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 141s 13s/step - accuracy: 0.5203 - loss: 1.0780 - val_accuracy: 0.5625 - val_loss: 0.8277
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 104s 13s/step - accuracy: 0.5663 - loss: 0.8218 - val_accuracy: 0.5625 - val_loss: 0.7485
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 145s 13s/step - accuracy: 0.7109 - loss: 0.6565 - val_accuracy: 0.7812 - val_loss: 0.4552
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 0.8080 - loss: 0.5166 - val_accuracy: 0.9062 - val_loss: 0.3929
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 0.8554 - loss: 0.3828 - val_accuracy: 0.9688 - val_loss: 0.1909
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 102s 13s/step - accuracy: 0.9320 - loss: 0.2421 - val_accuracy: 0.9062 - val_loss: 0.2162
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 100s 12s/step - accuracy: 0.9728 - loss: 0.1459 - val_accuracy: 0.9688 - val_loss: 0.1431
Epoch 10/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 102s 13s/step - accuracy: 0.9830 - loss: 0.1093 - val_accuracy: 1.0000 - val_loss: 0.0679
Epoch 11/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 104s 13s/step - accuracy: 0.9872 - loss: 0.0752 - val_accuracy: 1.0000 - val_loss: 0.0454
Epoch 12/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 104s 13s/step - accuracy: 0.9991 - loss: 0.0389 - val_accuracy: 1.0000 - val_loss: 0.0412
Epoch 13/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 103s 13s/step - accuracy: 0.9991 - loss: 0.0307 - val_accuracy: 0.9688 - val_loss: 0.0505
Epoch 14/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 1.0000 - loss: 0.0133 - val_accuracy: 1.0000 - val_loss: 0.0133
Epoch 15/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 100s 13s/step - accuracy: 1.0000 - loss: 0.0130 - val_accuracy: 1.0000 - val_loss: 0.0153
```

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

plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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




```
model.save('BestModel_VGG-16_SQLAlchemy.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
```

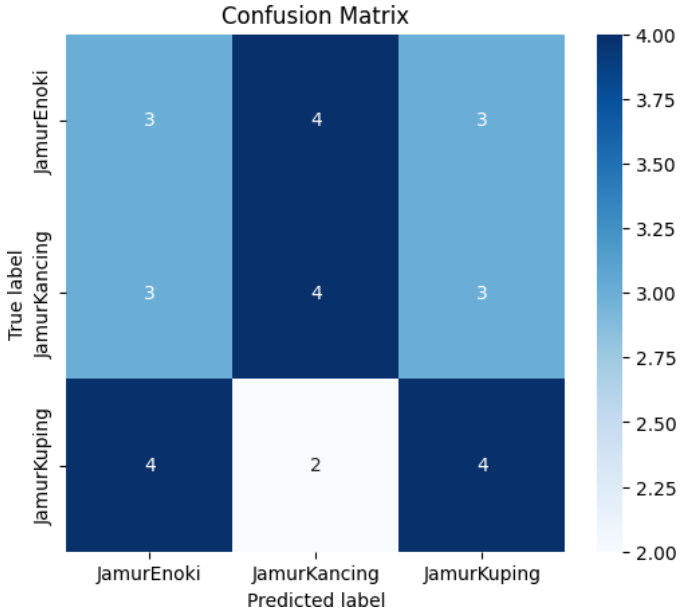
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

model = load_model(r'BestModel_VGG-16_SQLAlchemy.h5')
class_names = ['JamurEnoki', 'JamurKancing', 'JamurKuping']

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

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

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

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

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

```
result = classify_images(r"test_data\JamurKancing\JamurKancingTest_07.jpg")
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 688ms/step
Prediksi: JamurKancing
Confidence: 57.55%
Prediksi: JamurKancing dengan confidence 57.55%. Gambar asli disimpan di predicted_image.jpg.
```

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


test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

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

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

conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class, num_classes=3)

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

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

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

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["JamurEnoki", "JamurKancing", "JamurKuping"], yticklabels=["JamurEnoki", "JamurKancing", "JamurKuping"])
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 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
```




```
Confusion Matrix:
 [[3 4 3]
 [3 4 3]
 [4 2 4]]
Akurasi: 0.36666666666666664
Presisi: [0.3 0.4 0.4]
Recall: [0.3 0.4 0.4]
F1 Score: [0.3 0.4 0.4]
```

Indentitas

Sebastian Willys Lambang/200710639

SQLAlchemy_KlasifikasiJamurEnokiKancingKuping_MobileNet

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

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

```
import os

count = 0
dirs = os.listdir(r'train_data')
for dir in dirs:
    folder_path = os.path.join('train_data', dir)
    if os.path.isdir(folder_path):
        files = os.listdir(folder_path)
        print(f"{dir} Folder has {len(files)} Images")
        count += len(files)
print(f"Images Folder has {count} Images")
```

```
JamurEnoki Folder has 100 Images
JamurKancing Folder has 100 Images
JamurKuping Folder has 100 Images
Images Folder has 300 Images
```

```
base_dir = r'train_data'
img_size = 180
batch = 32
validation_split = 0.1

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.
```

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

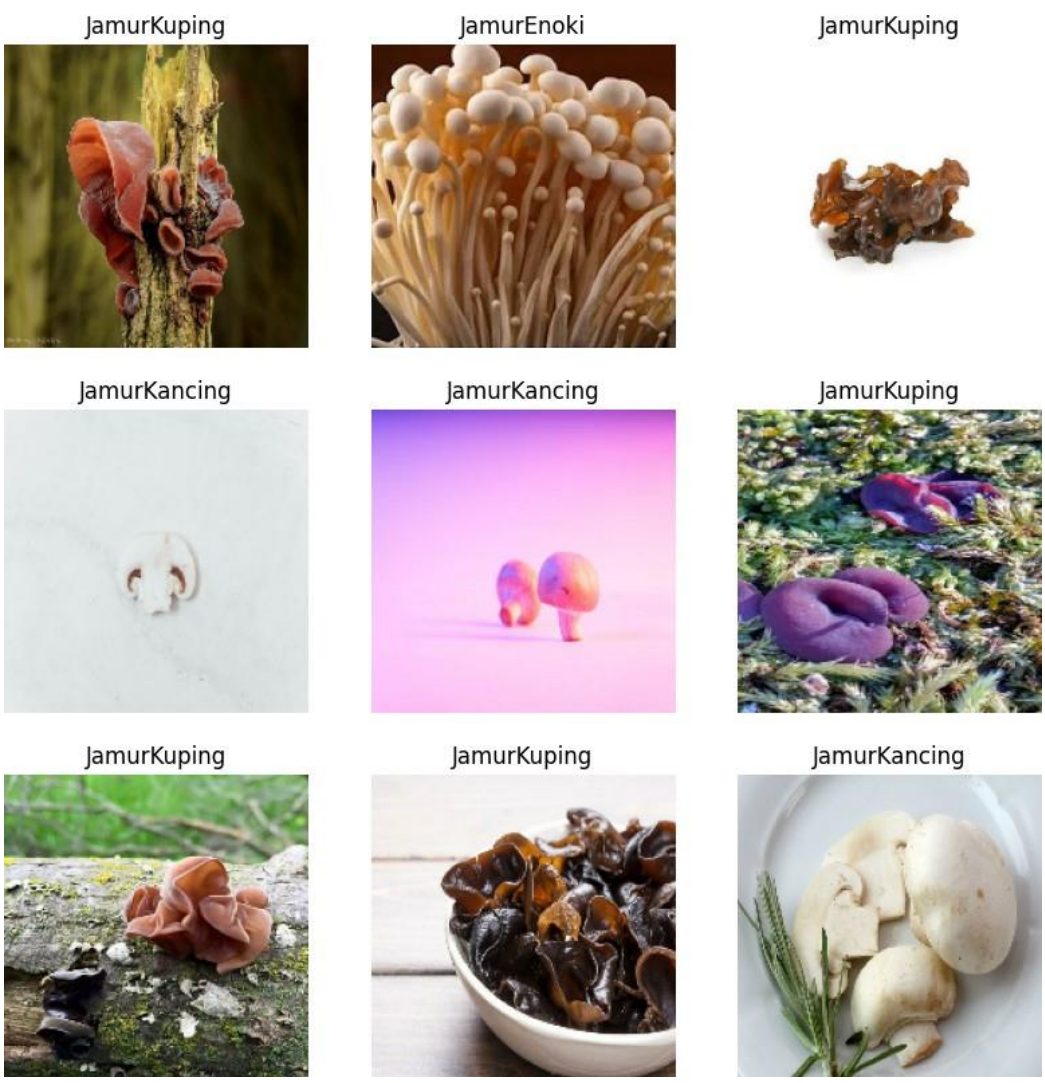
```
Class Names: ['JamurEnoki', 'JamurKancing', 'JamurKuping']
```

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

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

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

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

```
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')
```




```
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)
```

```
AUTOTUNE = tf.data.AUTOTUNE


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


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


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

```
c:\Python39\lib\site-packages\keras\src\layers\preprocessing\tf_data_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
```

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

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

```
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model


base_model = MobileNet(include_top=False, input_shape=(img_size, img_size, 3))

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

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

    ])
```

```
C:\Users\Antonius Rio\AppData\Local\Temp\ipykernel_18412\99726284.py:5: UserWarning: `input_shape` is undefined or non-square, or `
  base_model = MobileNet(include_top=False, input_shape=(img_size, img_size, 3))
```

```
from tensorflow.keras.optimizers import Adam

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



model.summary()
```

```
Model: "sequential_1"
```

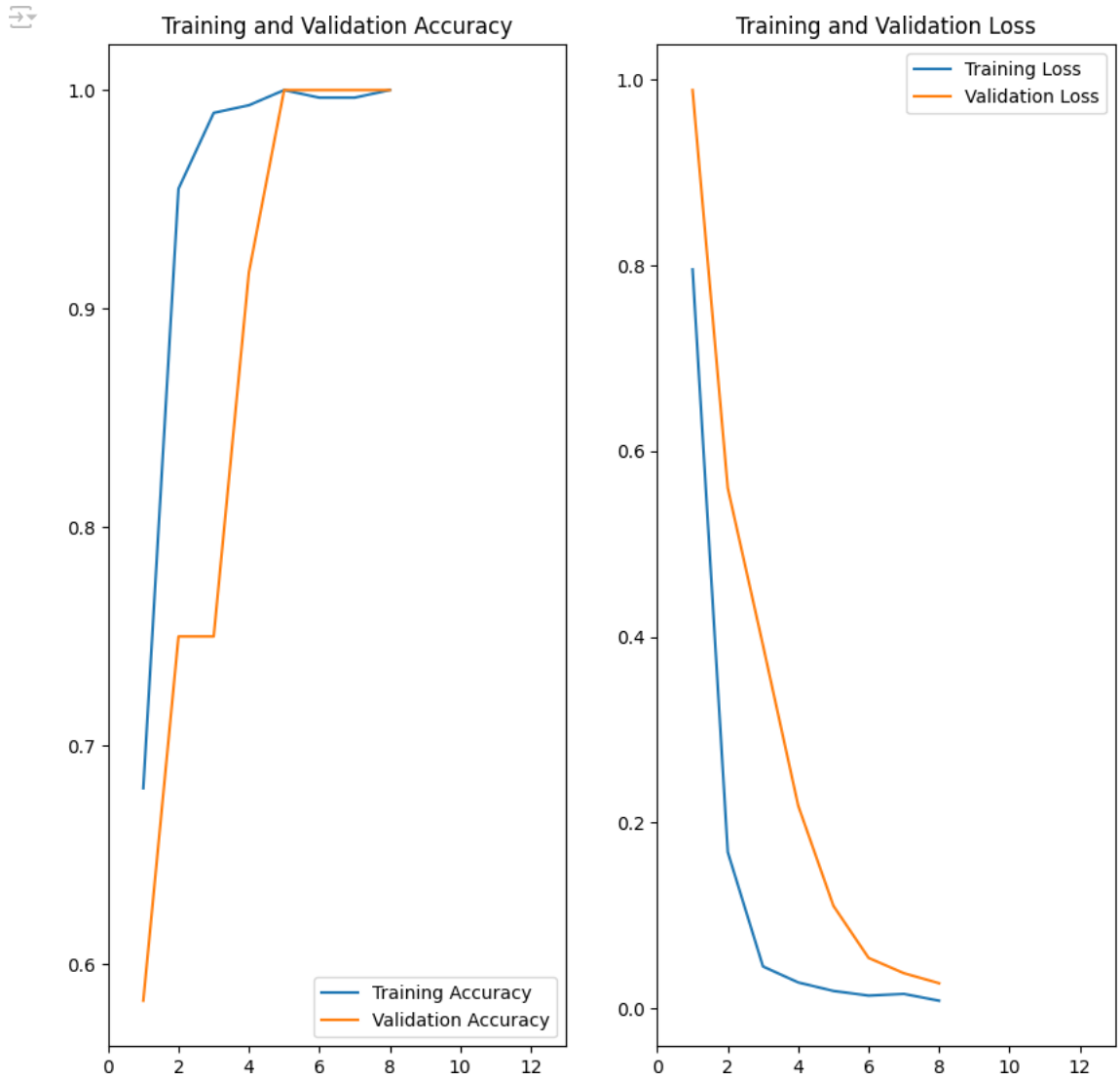
Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 180, 180, 3)	0
rescaling (Rescaling)	(None, 180, 180, 3)	0
mobilenet_1.00_224 (Functional)	(None, 5, 5, 1024)	3,228,864
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1024)	0
dense (Dense)	(None, 128)	131,200
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

```
Total params: 3,360,451 (12.82 MB)
Trainable params: 3,069,443 (11.71 MB)
Non-trainable params: 291 008 (1 11 MB)
```

```
from tensorflow.keras.callbacks import EarlyStopping

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

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

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 31s 2s/step - accuracy: 0.5685 - loss: 1.0810 - val_accuracy: 0.5833 - val_loss: 0.9887
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 12s 1s/step - accuracy: 0.9449 - loss: 0.1882 - val_accuracy: 0.7500 - val_loss: 0.5603
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.9860 - loss: 0.0619 - val_accuracy: 0.7500 - val_loss: 0.3909
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.9901 - loss: 0.0296 - val_accuracy: 0.9167 - val_loss: 0.2183
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 987ms/step - accuracy: 1.0000 - loss: 0.0200 - val_accuracy: 1.0000 - val_loss: 0.1104
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.9908 - loss: 0.0217 - val_accuracy: 1.0000 - val_loss: 0.0543
Epoch 7/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9939 - loss: 0.0225 - val_accuracy: 1.0000 - val_loss: 0.0379
Epoch 8/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 1.0000 - loss: 0.0075 - val_accuracy: 1.0000 - val_loss: 0.0271
```

```
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()
```




```
model.save('BestModel_MobileNet_SQLAlchemy.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
```

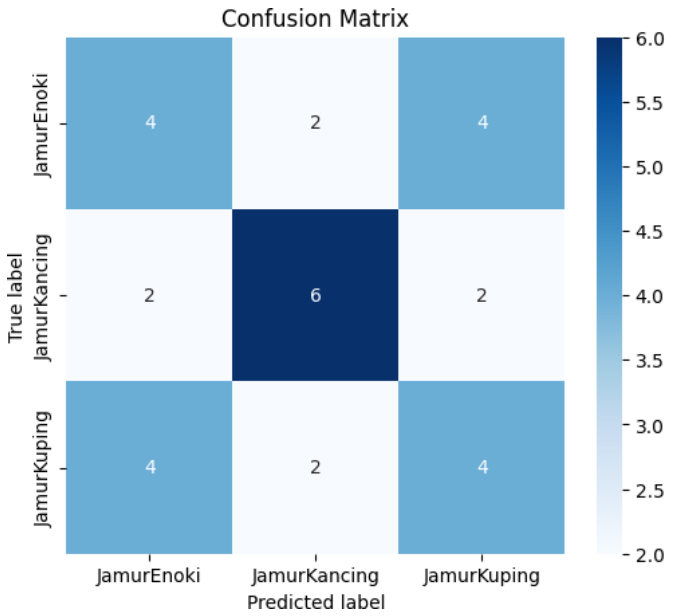
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from PIL import Image

model = load_model(r'BestModel_MobileNet_SQLAlchemy.h5')
class_names = ['JamurEnoki', 'JamurKancing', 'JamurKuping']

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

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

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

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

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

```
result = classify_images(r'test_data/JamurKancing/JamurKancingTest_01.jpg', save_path='JamurKancing.jpg')
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 902ms/step
Prediksi: JamurKancing
Confidence: 57.61%
Prediksi: JamurKancing dengan confidence 57.61%. Gambar asli disimpan di JamurKancing.jpg.
```

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

mobileNet_model = load_model(r'BestModel_MobileNet_SQLAlchemy.h5')

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

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

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

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

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

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

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

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

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

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```




```
Confusion Matrix:
 [[4 2 4]
 [2 6 2]
 [4 2 4]]
Akurasi: 0.4666666666666667
Presisi: [0.4 0.6 0.4]
Recall: [0.4 0.6 0.4]
F1 Score: [0 4 0 6 0 4]
```

Indentitas

Antonius Rosarianto Wisnu Putro/200710813

Jhonatan Emanuel Wangge/210711294

SQLAlchemy_KlasifikasiJamurEnokiKancingKuping_GoogleNet

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

data_dir = r"train_data"

data = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(180, 180),
    batch_size=None
)

print(data.class_names)

class_names = data.class_names
img_size = 180
batch = 32
validation_split = 0.1

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

total_count = len(list(dataset)) * batch
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 // batch)
```

```
val_ds = dataset.skip(train_count // batch)
```

```
Found 300 files belonging to 3 classes.
['JamurEnoki', 'JamurKancing', 'JamurKuping']
Found 300 files belonging to 3 classes.
Total Images: 320
Train Images: 288
Validation Images: 32
```

```
import matplotlib.pyplot as plt

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

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

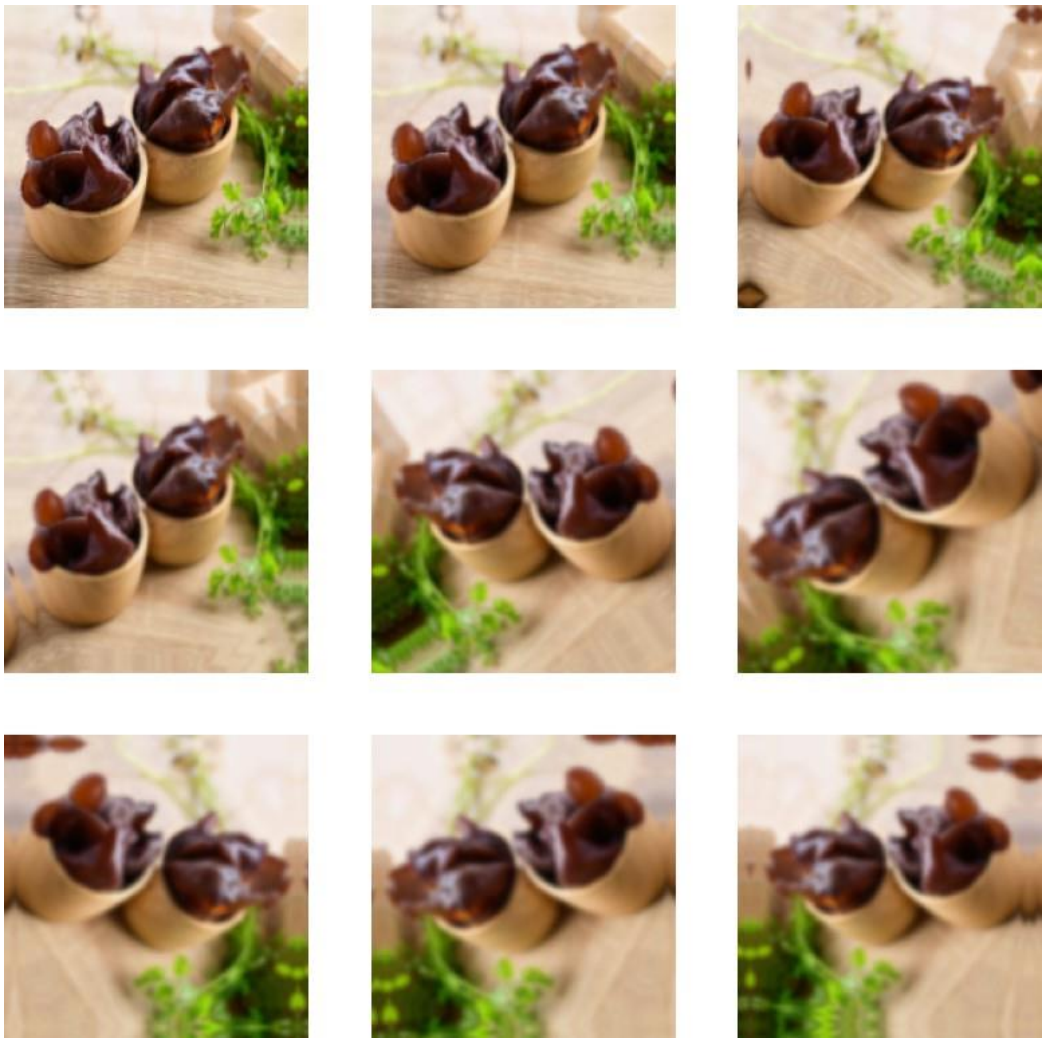

```
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape)
```

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

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

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

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

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

i = 0
plt.figure(figsize=(10,10))
for images, labels in train_ds.take(69):
    for i in range(9):
        images = data_augmentation(images)
        plt.subplot(3,3, i+1)
        plt.imshow(images[0].numpy().astype('uint8'))
        plt.axis('off')
```

```
c:\Python39\lib\site-packages\keras\src\layers\preprocessing\tf_data_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
```



```
import tensorflow as tf
import keras

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

from keras._tf_keras.keras.models import load_model

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()
```

WARNING:tensorflow:From c:\Python39\lib\site-packages\keras\src\backend\common\global_state.py:82: The name tf.reset_default_graph

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 180, 180, 3)	0	-
conv2d (Conv2D)	(None, 90, 90, 64)	9,472	input_layer[0][0]
max_pooling2d (MaxPooling2D)	(None, 45, 45, 64)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 45, 45, 64)	4,160	max_pooling2d[0]…
conv2d_2 (Conv2D)	(None, 45, 45, 192)	110,784	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 22, 22, 192)	0	conv2d_2[0][0]
conv2d_4 (Conv2D)	(None, 22, 22, 96)	18,528	max_pooling2d_1[…
conv2d_6 (Conv2D)	(None, 22, 22, 16)	3,088	max_pooling2d_1[…
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 192)	0	max_pooling2d_1[…
conv2d_3 (Conv2D)	(None, 22, 22, 64)	12,352	max_pooling2d_1[…
conv2d_5 (Conv2D)	(None, 22, 22, 128)	110,720	conv2d_4[0][0]
conv2d_7 (Conv2D)	(None, 22, 22, 32)	12,832	conv2d_6[0][0]
conv2d_8 (Conv2D)	(None, 22, 22, 32)	6,176	max_pooling2d_2[…
concatenate (Concatenate)	(None, 22, 22, 256)	0	conv2d_3[0][0], conv2d_5[0][0], conv2d_7[0][0], conv2d_8[0][0]
conv2d_10 (Conv2D)	(None, 22, 22, 128)	32,896	concatenate[0][0]
conv2d_12 (Conv2D)	(None, 22, 22, 32)	8,224	concatenate[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 22, 22, 256)	0	concatenate[0][0]
conv2d_9 (Conv2D)	(None, 22, 22, 128)	32,896	concatenate[0][0]
conv2d_11 (Conv2D)	(None, 22, 22, 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,	179,920	conv2d_16[0][0]

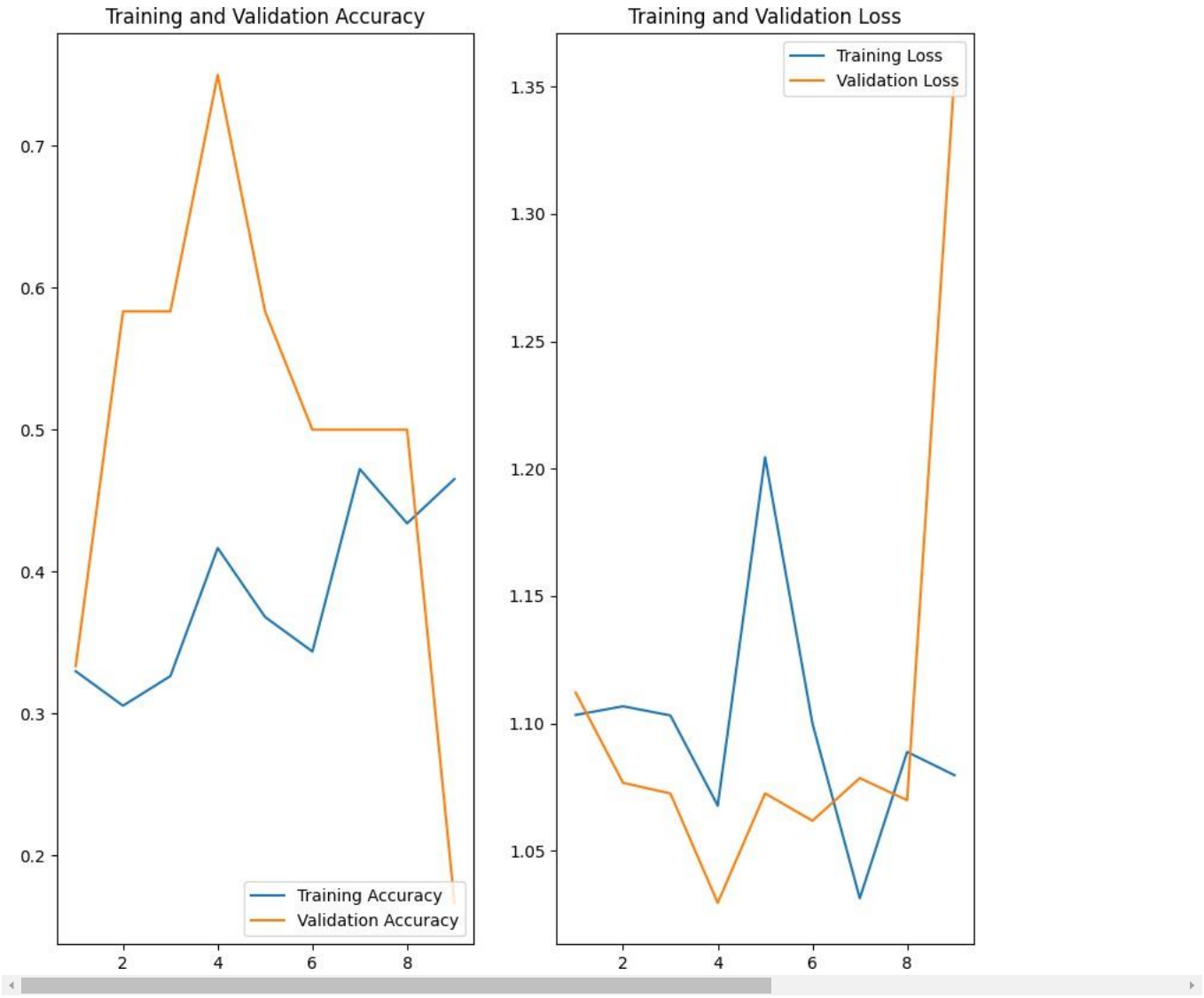
	208)		
conv2d_19 (Conv2D)	(None, 11, 11, 48)	19,248	conv2d_18[0][0]
conv2d_20 (Conv2D)	(None, 11, 11, 64)	30,784	max_pooling2d_5[…
concatenate_2 (Concatenate)	(None, 11, 11, 512)	0	conv2d_15[0][0], conv2d_17[0][0], conv2d_19[0][0], conv2d_20[0][0]
conv2d_22 (Conv2D)	(None, 11, 11, 112)	57,456	concatenate_2[0]…
conv2d_24 (Conv2D)	(None, 11, 11, 24)	12,312	concatenate_2[0]…
max_pooling2d_6 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_2[0]…
conv2d_21 (Conv2D)	(None, 11, 11, 160)	82,080	concatenate_2[0]…
conv2d_23 (Conv2D)	(None, 11, 11, 224)	226,016	conv2d_22[0][0]
conv2d_25 (Conv2D)	(None, 11, 11, 64)	38,464	conv2d_24[0][0]
conv2d_26 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_6[…
concatenate_3 (Concatenate)	(None, 11, 11, 512)	0	conv2d_21[0][0], conv2d_23[0][0], conv2d_25[0][0], conv2d_26[0][0]
conv2d_28 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]…
conv2d_30 (Conv2D)	(None, 11, 11, 24)	12,312	concatenate_3[0]…
max_pooling2d_7 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_3[0]…
conv2d_27 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]…
conv2d_29 (Conv2D)	(None, 11, 11, 256)	295,168	conv2d_28[0][0]
conv2d_31 (Conv2D)	(None, 11, 11, 64)	38,464	conv2d_30[0][0]
conv2d_32 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_7[…
concatenate_4 (Concatenate)	(None, 11, 11, 512)	0	conv2d_27[0][0], conv2d_29[0][0], conv2d_31[0][0], conv2d_32[0][0]
conv2d_34 (Conv2D)	(None, 11, 11, 144)	73,872	concatenate_4[0]…
conv2d_36 (Conv2D)	(None, 11, 11, 32)	16,416	concatenate_4[0]…
max_pooling2d_8 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_4[0]…
conv2d_33 (Conv2D)	(None, 11, 11, 112)	57,456	concatenate_4[0]…
conv2d_35 (Conv2D)	(None, 11, 11, 288)	373,536	conv2d_34[0][0]
conv2d_37 (Conv2D)	(None, 11, 11, 64)	51,264	conv2d_36[0][0]
conv2d_38 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_8[…
concatenate_5 (Concatenate)	(None, 11, 11, 528)	0	conv2d_33[0][0], conv2d_35[0][0], conv2d_37[0][0], conv2d_38[0][0]
conv2d_40 (Conv2D)	(None, 11, 11,	84,640	concatenate_5[0]…

	160)		
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	(None, 4, 4,	0	concatenate_8[0]…

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

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

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

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

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 38s 2s/step - accuracy: 0.3372 - loss: 1.1049 - val_accuracy: 0.3333 - val_loss: 1.1121
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.3465 - loss: 1.0988 - val_accuracy: 0.5833 - val_loss: 1.0767
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.3262 - loss: 1.1026 - val_accuracy: 0.5833 - val_loss: 1.0725
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 16s 2s/step - accuracy: 0.3578 - loss: 1.0850 - val_accuracy: 0.7500 - val_loss: 1.0296
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 0.4164 - loss: 1.2642 - val_accuracy: 0.5833 - val_loss: 1.0725
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 21s 2s/step - accuracy: 0.3409 - loss: 1.1023 - val_accuracy: 0.5000 - val_loss: 1.0618
Epoch 7/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 0.4708 - loss: 1.0281 - val_accuracy: 0.5000 - val_loss: 1.0785
Epoch 8/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 0.4766 - loss: 1.0848 - val_accuracy: 0.5000 - val_loss: 1.0698
Epoch 9/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.5306 - loss: 1.0553 - val_accuracy: 0.1667 - val_loss: 1.3546
```

```
ephocs_range = range(1, len(history.history['loss']) + 1)
plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.plot(ephocs_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(ephocs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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




```
model.save('BestModel_GoogleNet_SQLAlchemy.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
```

```
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'BestModel_GoogleNet_SQLAlchemy.h5')
class_names = ['JamurEnoki', 'JamurKancing', 'JamurKuping']
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('test_data/JamurKancing/JamurKancingTest_01.jpg', save_path='JamurKancing.jpg')
```

```
print(result)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 904ms/step
Prediksi: JamurKancing
Confidence: 38.12%
Prediksi: JamurKancing dengan confidence 38.12%. Gambar asli disimpan di JamurKancing.jpg.
```

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

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

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

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

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

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

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

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

plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["JamurEnoki", "JamurKancing", "JamurKuping"], yticklabels=["JamurEnoki", "JamurKancing", "JamurKuping"])
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
```