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

# Memuat model
model = load_model(r'D:\TUBES PMDPM\mobilenet\model_mobilenet.h5')
class_names = ['Busuk', 'Matang', 'Mentah']

def classify_image(image_path):
    try:
        # Memuat gambar dan mengubah ukurannya sesuai dengan model
        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)

        # Prediksi menggunakan model
        predictions = model.predict(input_image_exp_dim)
        result = tf.nn.softmax(predictions[0])  # Softmax untuk mendapatkan probabilitas

        # Mendapatkan kelas dengan confidence tertinggi
        class_idx = np.argmax(result)
        confidence_scores = result.numpy()
        return class_names[class_idx], confidence_scores
    except Exception as e:
        return "Error", str(e)
```

```
def display_progress_bar(confidence, class_names):
    for i, class_name in enumerate(class_names):
        percentage = confidence[i] * 100
        st.sidebar.progress(int(percentage))  # Streamlit progress bar
        st.sidebar.write(f"{class_name}: {percentage:.2f}%")

# Streamlit UI
st.title("Prediksi Kematangan Buah Rambutan")

uploaded_files = st.file_uploader("Unggah Gambar (Beberapa diperbolehkan) dan pastikan gambar sudah terupload dengan benar, dan jika terjadi kesalahan refresh saja halamannya. terima kasih.", type=["jpg", "png", "jpeg"], accept_multiple_files=True)

if st.sidebar.button("Prediksi"):
    if uploaded_files:
        st.sidebar.write("### Hasil Prediksi")
        for uploaded_file in uploaded_files:
            with open(uploaded_file.name, "wb") as f:
                f.write(uploaded_file.getbuffer())

            label, confidence = classify_image(uploaded_file.name)

            if label != "Error":
                st.sidebar.write(f"*Prediksi: {label}*")
                st.sidebar.write("Confidence:")
                display_progress_bar(confidence, class_names)
```

```
            st.sidebar.write("---")
        else:
                      st.sidebar.error(f"Kesalahan saat memproses gambar {uploaded_file.name}: {confidence}")
    else:
        st.sidebar.error("Silahkan unggah setidaknya satu gambar untuk diprediksi")


if uploaded_files:
    st.write("### Preview Gambar")
    for uploaded_file in uploaded_files:
        image = Image.open(uploaded_file)
        st.image(image, caption=f"{uploaded_file.name}", use_column_width=True)
```

Nama : Yohanes Paulus Ardha Suban Koten NPM : 220711658 Kelompok SB : Transformers Topik : Klasifikasi Kematangan Rambutan Arsitektur : AlexNet

```
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
import matplotlib.pyplot as plt

base_dir = r'C:\Users\ACER\Downloads\AlexNet_A_Transformers_Ardha\
train_data'
img_size = 180
batch = 32
validation_split = 0.1

val_ds = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
    validation_split=validation_split,
    subset="validation"
)
```

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

```
Using 30 files for validation.
```

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

```
Class Names: ['Busuk', 'Matang', 'Mentah']
```

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
    validation_split=validation_split,
    subset="training"
).map(lambda x, y: (data_augmentation(x, training=True), y)) \
 .cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
```

```
Found 300 files belonging to 3 classes.
Using 270 files for training.
```

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

```python
model = Sequential([
    layers.Rescaling(1./255, input_shape=(img_size, img_size, 3)),  # Normalization only
```

```
    # Convolutional Layers
    layers.Conv2D(96, (11, 11), strides=4, activation='relu'),
    layers.MaxPooling2D((3, 3), strides=2),
    layers.BatchNormalization(),

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

    layers.Conv2D(384, (3, 3), activation='relu', padding='same'),
    layers.Conv2D(384, (3, 3), activation='relu', padding='same'),
    layers.Conv2D(256, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((3, 3), strides=2),
    layers.BatchNormalization(),

    # Flatten and Fully Connected Layers
    layers.Flatten(),
    layers.Dense(4096, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(4096, activation='relu'),
    layers.Dropout(0.5),

    # Output Layer
    layers.Dense(len(class_names), activation='softmax')
])
```

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

```
from tensorflow.keras.optimizers import Adam

model.compile(
    optimizer=Adam(learning_rate=1e-5),  # Reduced learning rate for
better convergence
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

```
Model: "sequential_4"

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

rescaling_2 (Rescaling)	(None, 180, 180, 3)	0
conv2d_10 (Conv2D)	(None, 43, 43, 96)	34,944
max_pooling2d_6 (MaxPooling2D)	(None, 21, 21, 96)	0
batch_normalization_6 (BatchNormalization)	(None, 21, 21, 96)	384
conv2d_11 (Conv2D)	(None, 21, 21, 256)	614,656
max_pooling2d_7 (MaxPooling2D)	(None, 10, 10, 256)	0
batch_normalization_7 (BatchNormalization)	(None, 10, 10, 256)	1,024
conv2d_12 (Conv2D)	(None, 10, 10, 384)	885,120
conv2d_13 (Conv2D)	(None, 10, 10, 384)	1,327,488
conv2d_14 (Conv2D)	(None, 10, 10, 256)	884,992
max_pooling2d_8 (MaxPooling2D)	(None, 4, 4, 256)	0

```
───────┤
│ batch_normalization_8                │ (None, 4, 4, 256)          │
1,024 │
│ (BatchNormalization)                 │                            │
│
├──────────────────────────────────────┼────────────────────────────┼──────────
───────┤
│ flatten_2 (Flatten)                  │ (None, 4096)               │
0 │
├──────────────────────────────────────┼────────────────────────────┼──────────
───────┤
│ dense_6 (Dense)                      │ (None, 4096)               │
16,781,312 │
├──────────────────────────────────────┼────────────────────────────┼──────────
───────┤
│ dropout_4 (Dropout)                  │ (None, 4096)               │
0 │
├──────────────────────────────────────┼────────────────────────────┼──────────
───────┤
│ dense_7 (Dense)                      │ (None, 4096)               │
16,781,312 │
├──────────────────────────────────────┼────────────────────────────┼──────────
───────┤
│ dropout_5 (Dropout)                  │ (None, 4096)               │
0 │
├──────────────────────────────────────┼────────────────────────────┼──────────
───────┤
│ dense_8 (Dense)                      │ (None, 3)                  │
12,291 │
└──────────────────────────────────────┴────────────────────────────┴──────────
───────┘

 Total params: 37,324,547 (142.38 MB)

 Trainable params: 37,323,331 (142.38 MB)

 Non-trainable params: 1,216 (4.75 KB)
```

```
history = model.fit(
    train_ds,
    epochs=30,
    validation_data=val_ds,
    validation_steps=len(val_ds) // batch,
)
```

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 8s 490ms/step - accuracy: 0.4187 - loss:
1.2972 - val_accuracy: 0.5667 - val_loss: 1.0928
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 461ms/step - accuracy: 0.6934 - loss:
```

```
0.7845 - val_accuracy: 0.5333 - val_loss: 1.0887
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 488ms/step - accuracy: 0.7577 - loss:
0.5353 - val_accuracy: 0.3000 - val_loss: 1.0867
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 480ms/step - accuracy: 0.8957 - loss:
0.3283 - val_accuracy: 0.2667 - val_loss: 1.0886
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 455ms/step - accuracy: 0.9614 - loss:
0.1894 - val_accuracy: 0.3333 - val_loss: 1.0853
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 458ms/step - accuracy: 0.8933 - loss:
0.2295 - val_accuracy: 0.5667 - val_loss: 1.0762
Epoch 7/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 471ms/step - accuracy: 0.9752 - loss:
0.1224 - val_accuracy: 0.7333 - val_loss: 1.0678
Epoch 8/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 450ms/step - accuracy: 0.9425 - loss:
0.1442 - val_accuracy: 0.7333 - val_loss: 1.0540
Epoch 9/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 455ms/step - accuracy: 0.9689 - loss:
0.0905 - val_accuracy: 0.6000 - val_loss: 1.0413
Epoch 10/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 454ms/step - accuracy: 0.9897 - loss:
0.0665 - val_accuracy: 0.6667 - val_loss: 1.0291
Epoch 11/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 468ms/step - accuracy: 0.9911 - loss:
0.0583 - val_accuracy: 0.6667 - val_loss: 1.0179
Epoch 12/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 461ms/step - accuracy: 0.9896 - loss:
0.0590 - val_accuracy: 0.7000 - val_loss: 1.0074
Epoch 13/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 463ms/step - accuracy: 0.9917 - loss:
0.0566 - val_accuracy: 0.6000 - val_loss: 0.9978
Epoch 14/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 464ms/step - accuracy: 0.9953 - loss:
0.0444 - val_accuracy: 0.6667 - val_loss: 0.9850
Epoch 15/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 486ms/step - accuracy: 0.9943 - loss:
0.0378 - val_accuracy: 0.6667 - val_loss: 0.9709
Epoch 16/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 479ms/step - accuracy: 0.9882 - loss:
0.0406 - val_accuracy: 0.6667 - val_loss: 0.9569
Epoch 17/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 484ms/step - accuracy: 1.0000 - loss:
0.0255 - val_accuracy: 0.6667 - val_loss: 0.9489
Epoch 18/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 483ms/step - accuracy: 1.0000 - loss:
0.0194 - val_accuracy: 0.6667 - val_loss: 0.9427
```

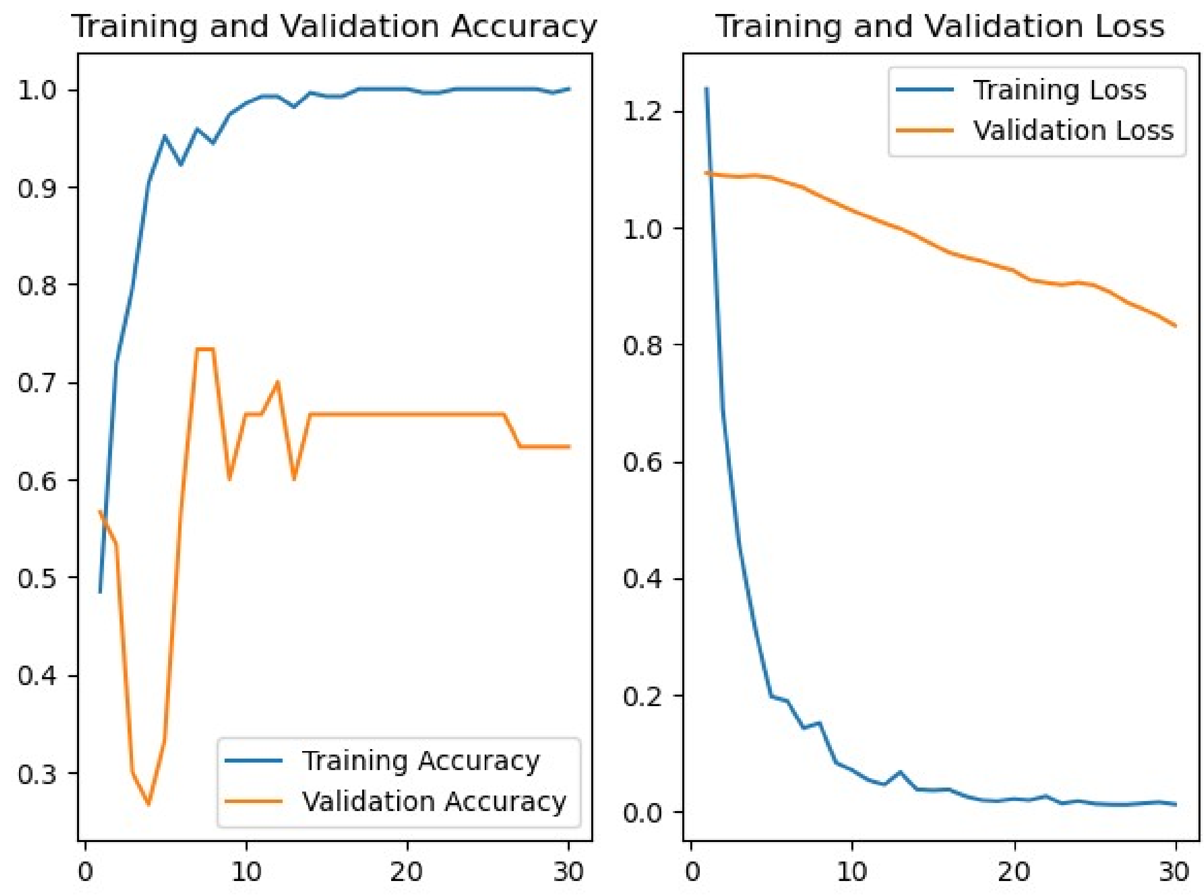
```
Epoch 19/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 472ms/step - accuracy: 1.0000 - loss:
0.0163 - val_accuracy: 0.6667 - val_loss: 0.9336
Epoch 20/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 467ms/step - accuracy: 1.0000 - loss:
0.0175 - val_accuracy: 0.6667 - val_loss: 0.9262
Epoch 21/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 462ms/step - accuracy: 0.9949 - loss:
0.0216 - val_accuracy: 0.6667 - val_loss: 0.9101
Epoch 22/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 461ms/step - accuracy: 0.9938 - loss:
0.0375 - val_accuracy: 0.6667 - val_loss: 0.9050
Epoch 23/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 459ms/step - accuracy: 1.0000 - loss:
0.0118 - val_accuracy: 0.6667 - val_loss: 0.9016
Epoch 24/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 463ms/step - accuracy: 1.0000 - loss:
0.0221 - val_accuracy: 0.6667 - val_loss: 0.9052
Epoch 25/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 455ms/step - accuracy: 1.0000 - loss:
0.0150 - val_accuracy: 0.6667 - val_loss: 0.9007
Epoch 26/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 478ms/step - accuracy: 1.0000 - loss:
0.0106 - val_accuracy: 0.6667 - val_loss: 0.8889
Epoch 27/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 467ms/step - accuracy: 1.0000 - loss:
0.0120 - val_accuracy: 0.6333 - val_loss: 0.8719
Epoch 28/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 468ms/step - accuracy: 1.0000 - loss:
0.0126 - val_accuracy: 0.6333 - val_loss: 0.8605
Epoch 29/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 478ms/step - accuracy: 0.9896 - loss:
0.0264 - val_accuracy: 0.6333 - val_loss: 0.8477
Epoch 30/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 4s 476ms/step - accuracy: 1.0000 - loss:
0.0134 - val_accuracy: 0.6333 - val_loss: 0.8321
```

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

plt.figure(figsize=(12, 4))
```

```
<Figure size 1200x400 with 0 Axes>

<Figure size 1200x400 with 0 Axes>
```

```python
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')

# Plot training & validation loss
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.tight_layout()
plt.show()
```




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

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

```
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        # Memuat dan mempersiapkan gambar untuk prediksi
        input_image = tf.keras.utils.load_img(image_path,
target_size=(img_size, img_size))  # Sesuaikan ukuran input
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_array = input_image_array / 255.0  # Normalisasi
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)  #
Menambahkan dimensi batch

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

        # Menampilkan hasil prediksi dan confidence
        print(f"Prediksi: {class_names[class_idx]}")
        print(f"Confidence: {confidence:.2f}%")

        # Menyimpan gambar asli tanpa teks
        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'C:\Users\ACER\Downloads\
AlexNet_A_Transformers_Ardha\test_data\Mentah\10.jpg',
save_path='mentah.jpg')
print(result)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step
Prediksi: Mentah
Confidence: 38.80%
Prediksi: Mentah dengan confidence 38.80%. Gambar asli disimpan di
mentah.jpg.
```

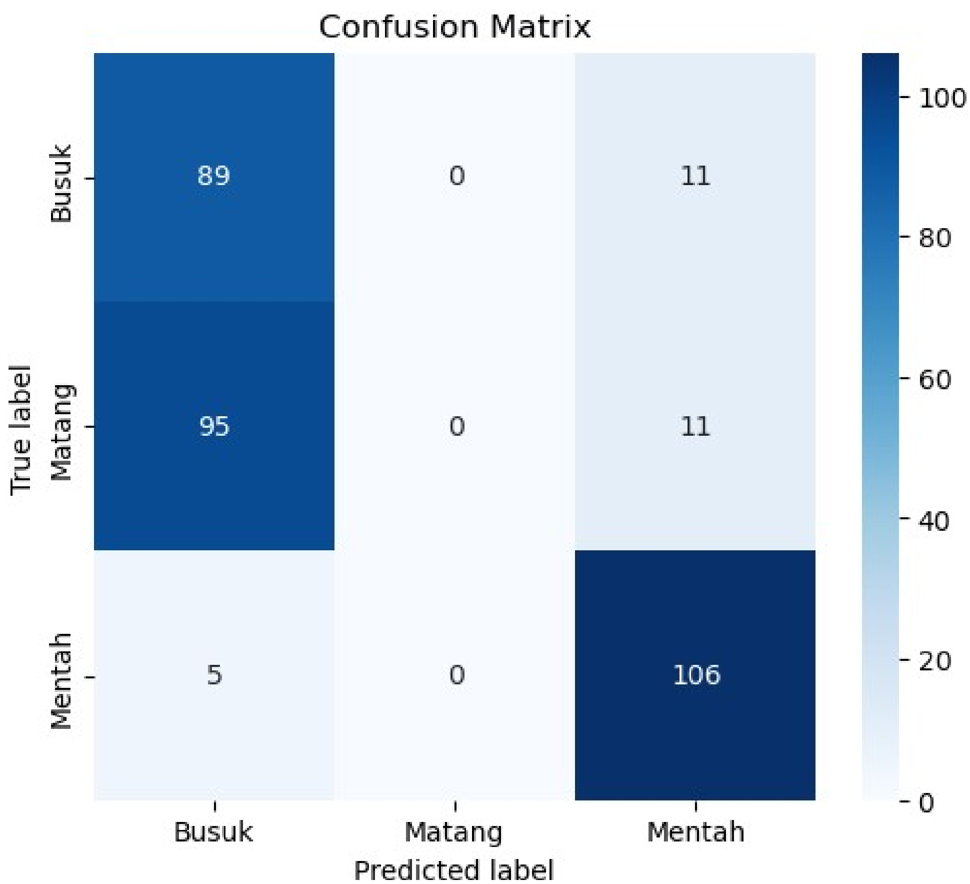
```
import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

# Load model MobileNet
mobileNet_model = load_model(r'C:\Users\ACER\Downloads\
AlexNet_A_Transformers_Ardha\model_AlexNet.h5')
class_names = ['Busuk', 'Matang', 'Mentah']
```

```
# Load test data
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180),
    shuffle=False  # Ensure order consistency for predictions
)

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

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

# Compute confusion matrix
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class,
num_classes=len(class_names))

# Compute metrics
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)

precision = tf.divide(
    tf.linalg.diag_part(conf_mat),
    tf.reduce_sum(conf_mat, axis=0),
    name="precision"
)
recall = tf.divide(
    tf.linalg.diag_part(conf_mat),
    tf.reduce_sum(conf_mat, axis=1),
    name="recall"
)
f1_score = tf.divide(
    2 * (precision * recall),
    precision + recall,
    name="f1_score"
)

# Handle NaNs in metrics
precision = tf.where(tf.math.is_nan(precision),
tf.zeros_like(precision), precision)
recall = tf.where(tf.math.is_nan(recall), tf.zeros_like(recall),
recall)
f1_score = tf.where(tf.math.is_nan(f1_score), tf.zeros_like(f1_score),
```

```
f1_score)

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

# Display metrics
print("Confusion Matrix:\n", conf_mat.numpy())
print("Accuracy:", accuracy.numpy())
print("Precision per class:", precision.numpy())
print("Recall per class:", recall.numpy())
print("F1 Score per class:", f1_score.numpy())
```

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

```
Found 317 files belonging to 3 classes.
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 109ms/step
```




```
Confusion Matrix:
 [[ 89   0  11]
 [ 95   0  11]
 [  5   0 106]]
Accuracy: 0.6151419558359621
Precision per class: [0.47089947 0.         0.828125  ]
Recall per class: [0.89       0.         0.95495495]
F1 Score per class: [0.61591696 0.         0.88702929]
```

Nama : Jonathan Louis Guido NPM : 220711660 Kelompok SB : Transformers Topik : Klasifikasi Kematangan Rambutan Arsitektur : MobileNet

```
import os
import numpy as np

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

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

```
Busuk Folder has 100 Images
Matang Folder has 100 Images
Mentah Folder has 100 Images
Images Folder has 300 Images
```

```
base_dir = r'D:\TUBES PMDPM\mobilenet\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: ['Busuk', 'Matang', 'Mentah']
```

```
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)
```

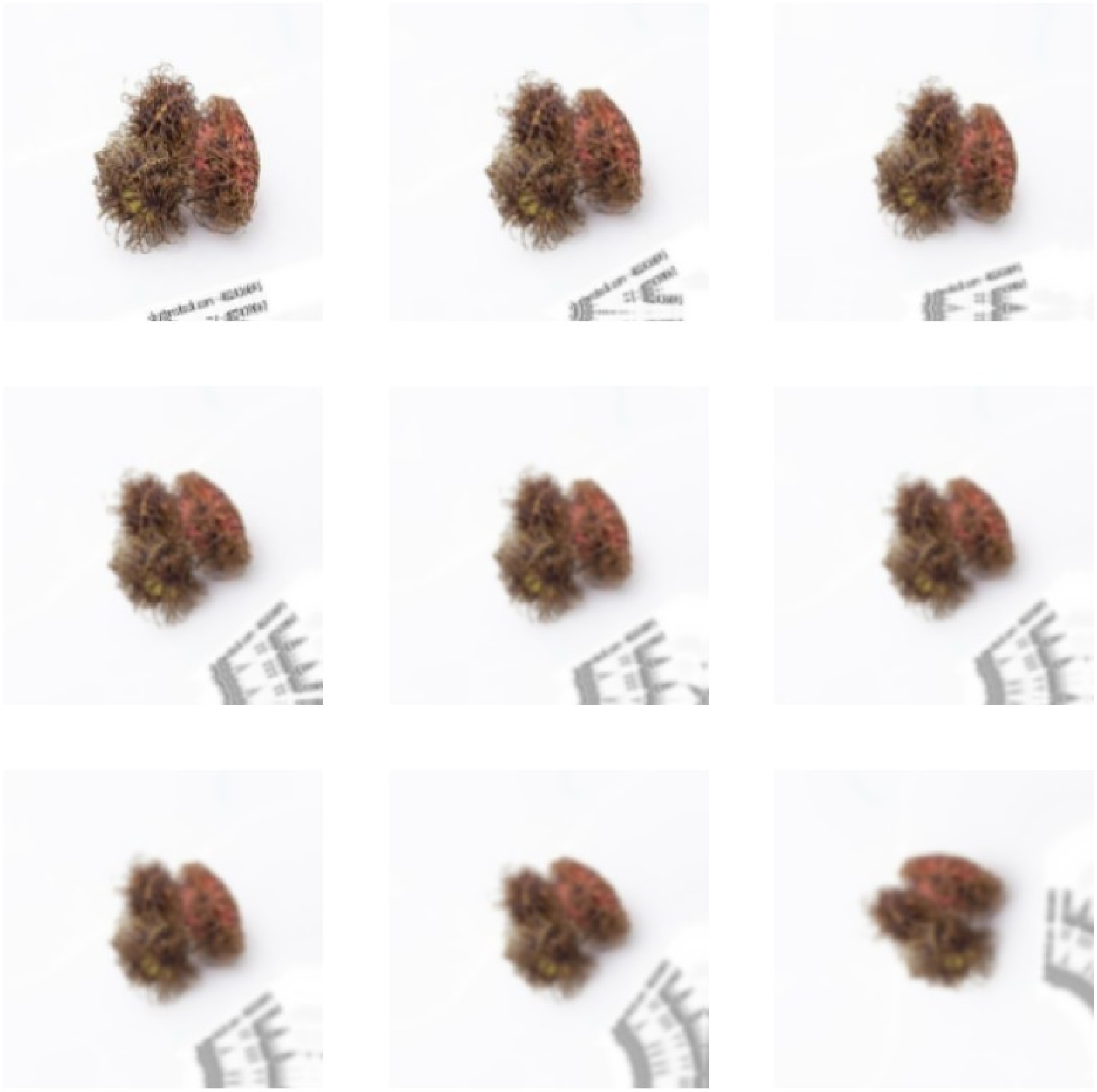
```
import matplotlib.pyplot as plt

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("diagonal", input_shape =
(img_size,img_size,3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])

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

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


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\ACER\AppData\Local\Temp\ipykernel_13976\2888943737.py:5:
UserWarning: `input_shape` is undefined or non-square, or `rows` is
not in [128, 160, 192, 224]. Weights for input shape (224, 224) will
be loaded as the default.
  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()
```

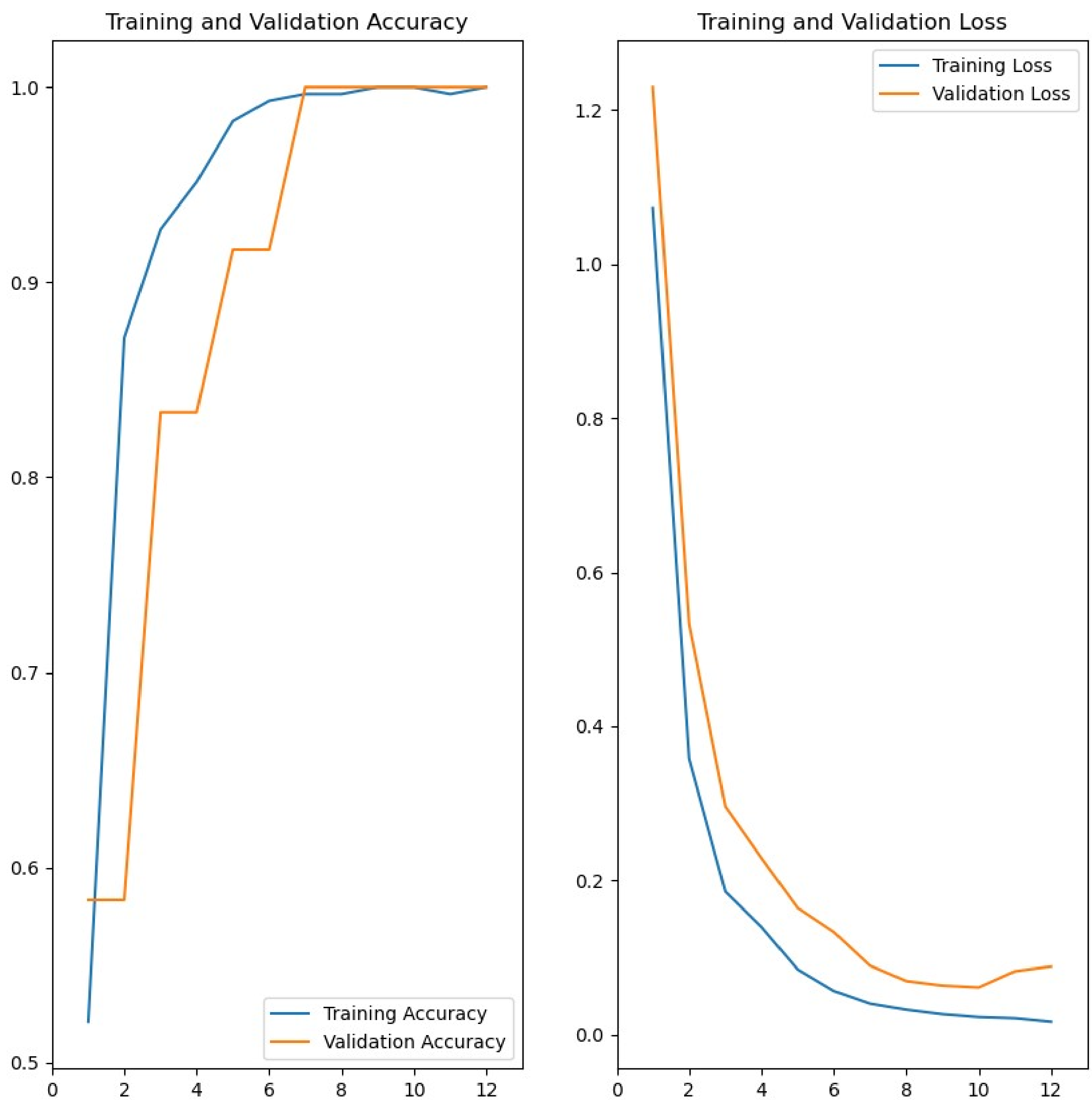
```
Model: "sequential_3"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ sequential_2 (Sequential)        │ (None, 180, 180, 3)     │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ rescaling_1 (Rescaling)          │ (None, 180, 180, 3)     │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ mobilenet_1.00_224 (Functional) │ (None, 5, 5, 1024)      │
3,228,864 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ global_average_pooling2d_1       │ (None, 1024)            │
0 │
│ (GlobalAveragePooling2D)         │                         │
```

```
│
├──────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ dense_2 (Dense)                  │ (None, 128)             │
131,200 │
├──────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ dropout_1 (Dropout)              │ (None, 128)             │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
───────┤
│ dense_3 (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=5,
                               mode='max')

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

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 50s 1s/step - accuracy: 0.4141 - loss: 1.3381
- val_accuracy: 0.5833 - val_loss: 1.2296
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.8333 - loss: 0.4287
- val_accuracy: 0.5833 - val_loss: 0.5334
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.9091 - loss: 0.2121
- val_accuracy: 0.8333 - val_loss: 0.2958
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9678 - loss: 0.1224
- val_accuracy: 0.8333 - val_loss: 0.2290
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9827 - loss: 0.0862
- val_accuracy: 0.9167 - val_loss: 0.1642
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.9913 - loss: 0.0539
- val_accuracy: 0.9167 - val_loss: 0.1330
```

```
Epoch 7/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.9993 - loss: 0.0420
- val_accuracy: 1.0000 - val_loss: 0.0893
Epoch 8/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.9973 - loss: 0.0312
- val_accuracy: 1.0000 - val_loss: 0.0692
Epoch 9/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 1.0000 - loss: 0.0231
- val_accuracy: 1.0000 - val_loss: 0.0634
Epoch 10/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 1.0000 - loss: 0.0203
- val_accuracy: 1.0000 - val_loss: 0.0610
Epoch 11/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.9955 - loss: 0.0248
- val_accuracy: 1.0000 - val_loss: 0.0819
Epoch 12/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 1.0000 - loss: 0.0171
- val_accuracy: 1.0000 - val_loss: 0.0882

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('model_mobilenet.h5')
```

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

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

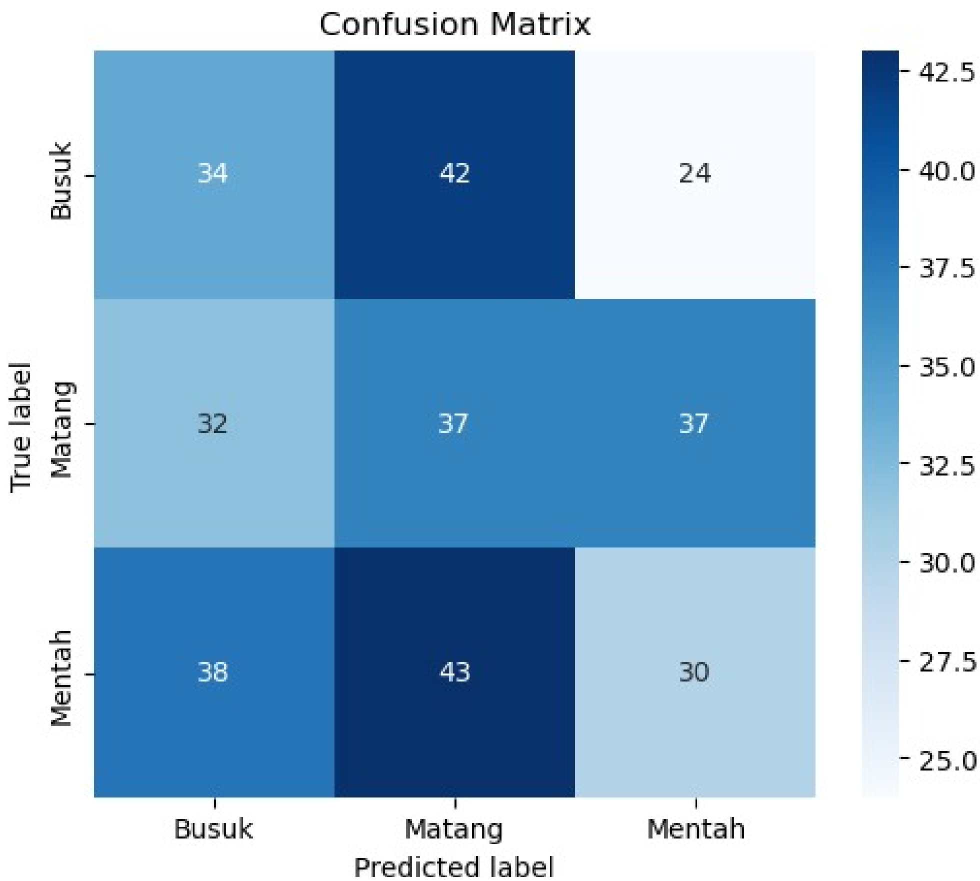
```
from PIL import Image

#memuat model yang sudah dilatih
model = load_model(r'D:\TUBES PMDPM\mobilenet\
BestModel_MobileNet_Transformers.h5')
class_names = ['Busuk', 'Matang', 'Mentah']

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


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

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

        #menyimpan gambar asli tanpa teks
        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:\TUBES PMDPM\mobilenet\test_data\Mentah\
2.jpg', save_path='mentah.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 you train
or evaluate the model.

1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
Prediksi: Mentah
Confidence: 53.35%
Prediksi: Mentah dengan confidence 53.35%. Gambar asli disimpan di
mentah.jpg.
```

```python
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'D:\TUBES PMDPM\mobilenet\
BestModel_MobileNet_Transformers.h5')
class_names = ['Busuk', 'Matang', 'Mentah']

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=["Busuk", "Matang", "Mentah"], yticklabels=["Busuk",
"Matang", "Mentah"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

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

```
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())
```

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

Found 317 files belonging to 3 classes.
10/10 ━━━━━━━━━━━━━━━━━━━━ 11s 766ms/step
```




```
Confusion Matrix:
 [[34 42 24]
 [32 37 37]
 [38 43 30]]
Akurasi: 0.3186119873817035
Presisi: [0.32692308 0.30327869 0.32967033]
Recall: [0.34       0.3490566  0.27027027]
F1 Score: [0.33333333 0.3245614  0.2970297 ]
```

#Dylan Hazael Raharja, 220711894, Transformers, Klasifikasi Kematangan Rambutan, dan mengerjakan VGG-16

#Eustakius Satu Rajawali Ku, 220711648, Transformers, Klasifikasi Kematangan Rambutan, dan mengerjakan VGG-16

```
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, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
Dropout, Flatten
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout, Conv2D,
MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import
image_dataset_from_directory

import matplotlib.pyplot as plt


base_dir = r"D:/ATMAJAYA/Semester5/PMDPL/UAS_A_11648/train_data"
img_size = 224  # VGG-16 menggunakan ukuran gambar 224x224
batch_size = 32
validation_split = 0.1

if os.path.exists(base_dir):
    print("Directory exists:", base_dir)
    print("Subdirectories:", os.listdir(base_dir))
else:
    raise FileNotFoundError(f"Directory not found: {base_dir}")
```

```
Directory exists: D:/ATMAJAYA/Semester5/PMDPL/UAS_A_11648/train_data
Subdirectories: ['Busuk', 'Matang', 'Mentah']
```

```
base_dir = r"D:/ATMAJAYA/Semester5/PMDPL/UAS_A_11648/train_data"

dataset = image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch_size
)
```

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

```
train_size = int(len(dataset) * 0.8)
val_size = int(len(dataset) * 0.1)
test_size = len(dataset) - train_size - val_size

train_ds = dataset.take(train_size)
val_ds = dataset.skip(train_size).take(val_size)
test_ds = dataset.skip(train_size + val_size)

import os

count = 0
dirs = os.listdir(r'D:\ATMAJAYA\Semester5\PMDPL\UAS_A_11648\
train_data')
for dir in dirs:
    files = list(os.listdir(os.path.join(r'D:\ATMAJAYA\Semester5\
PMDPL\UAS_A_11648\train_data', dir)))
    print(f'{dir} Folder has {len(files)} Images')
    count += len(files)
print(f'Images Folder has {count} Images')

Busuk Folder has 83 Images
Matang Folder has 106 Images
Mentah Folder has 111 Images
Images Folder has 300 Images

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

Class Names: ['Busuk', 'Matang', 'Mentah']

from tensorflow.keras import layers

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

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

import tensorflow as tf

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

```
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout, BatchNormalization

def fine_tuned_vgg16(input_shape, num_classes):
    model = Sequential([
        # Block 1
        Conv2D(64, (3, 3), activation='relu', padding='same',
input_shape=input_shape),
        BatchNormalization(),
        Conv2D(64, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D(pool_size=(2, 2)),

        # Block 2
        Conv2D(128, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(128, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D(pool_size=(2, 2)),

        # Block 3
        Conv2D(256, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(256, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(256, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D(pool_size=(2, 2)),

        # Block 4
        Conv2D(512, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(512, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(512, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D(pool_size=(2, 2)),

        # Block 5
        Conv2D(512, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(512, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(512, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D(pool_size=(2, 2)),
```

```
        # Fully Connected Layers
        Flatten(),
        Dense(1024, activation='relu'),  # Reduced from 4096 to 1024
        Dropout(0.5),
        Dense(512, activation='relu'),  # Further reduction to avoid 
overfitting
        Dropout(0.5),
        Dense(num_classes, activation='softmax')  # Adjust output 
layer for your number of classes
    ])
    return model


from tensorflow.keras.optimizers import Adam

input_shape = (224, 224, 3)
num_classes = 3

model = build_vgg16(input_shape, num_classes)

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

print("Model compiled successfully.")
```

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

```
Model compiled successfully.
```

```
import os

data_dir = r'D:\ATMAJAYA\Semester5\PMDPL\UAS_A_11648\train_data'
class_names = sorted(os.listdir(data_dir))  # Ensure alphabetical 
order for consistent class indices
print(f"Class Names: {class_names}")

input_shape = (img_size, img_size, 3)
num_classes = len(class_names)
model = build_vgg16(input_shape, num_classes)
```

```
print(f"Input Shape: {input_shape}")
print(f"Number of Classes: {num_classes}")
```

```
Class Names: ['Busuk', 'Matang', 'Mentah']
Input Shape: (224, 224, 3)
Number of Classes: 3
```

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3,
mode='max')
```

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

# Ensure model is built
input_shape = (img_size, img_size, 3)  # Define input shape
num_classes = len(class_names)         # Define number of classes
model = build_vgg16(input_shape, num_classes)

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

# Set up early stopping
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

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

```
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 107s 12s/step - accuracy: 0.2838 - loss:
2.3209 - val_accuracy: 0.3125 - val_loss: 1.1016
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 95s 12s/step - accuracy: 0.3477 - loss:
1.0992 - val_accuracy: 0.3125 - val_loss: 1.0474
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 96s 12s/step - accuracy: 0.5150 - loss:
0.9762 - val_accuracy: 0.8438 - val_loss: 0.3703
Epoch 4/30
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 95s 12s/step - accuracy: 0.7798 - loss:
0.6181 - val_accuracy: 0.5938 - val_loss: 0.9157
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 96s 12s/step - accuracy: 0.7299 - loss:
0.6952 - val_accuracy: 1.0000 - val_loss: 0.3221
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 91s 11s/step - accuracy: 0.8774 - loss:
0.3518 - val_accuracy: 1.0000 - val_loss: 0.0389
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 91s 11s/step - accuracy: 0.9329 - loss:
0.1968 - val_accuracy: 0.6562 - val_loss: 1.3125
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 90s 11s/step - accuracy: 0.8494 - loss:
0.4369 - val_accuracy: 0.9375 - val_loss: 0.2437
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 92s 12s/step - accuracy: 0.9387 - loss:
0.2141 - val_accuracy: 1.0000 - val_loss: 0.0290
Epoch 10/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 91s 11s/step - accuracy: 0.9548 - loss:
0.1144 - val_accuracy: 1.0000 - val_loss: 0.0387
Epoch 11/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 92s 12s/step - accuracy: 0.8850 - loss:
0.3014 - val_accuracy: 1.0000 - val_loss: 0.0461
Epoch 12/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 92s 11s/step - accuracy: 0.9605 - loss:
0.1637 - val_accuracy: 1.0000 - val_loss: 0.1681
Epoch 13/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 91s 11s/step - accuracy: 0.9536 - loss:
0.1402 - val_accuracy: 1.0000 - val_loss: 0.0101
Epoch 14/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 91s 11s/step - accuracy: 0.9533 - loss:
0.0741 - val_accuracy: 1.0000 - val_loss: 0.0066
Epoch 15/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 91s 11s/step - accuracy: 0.9840 - loss:
0.0346 - val_accuracy: 1.0000 - val_loss: 8.0506e-05
Epoch 16/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 90s 11s/step - accuracy: 0.9772 - loss:
0.0687 - val_accuracy: 1.0000 - val_loss: 0.0257
Epoch 17/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 90s 11s/step - accuracy: 0.9811 - loss:
0.0572 - val_accuracy: 1.0000 - val_loss: 0.0041
Epoch 18/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 90s 11s/step - accuracy: 0.9808 - loss:
0.0837 - val_accuracy: 1.0000 - val_loss: 0.0027
Epoch 19/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 90s 11s/step - accuracy: 0.9823 - loss:
0.0480 - val_accuracy: 1.0000 - val_loss: 0.0541
Epoch 20/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 92s 12s/step - accuracy: 0.9862 - loss:
```

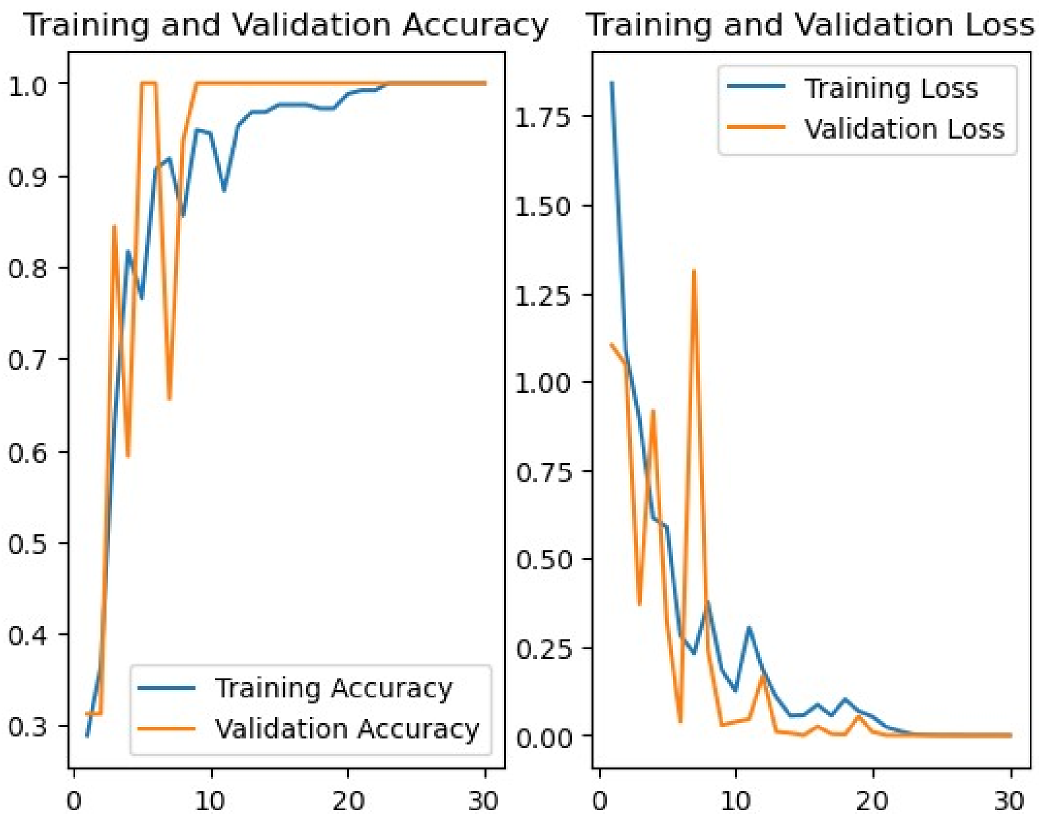
```
0.0746 - val_accuracy: 1.0000 - val_loss: 0.0100
Epoch 21/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 99s 13s/step - accuracy: 0.9888 - loss:
0.0435 - val_accuracy: 1.0000 - val_loss: 4.5475e-04
Epoch 22/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 115s 14s/step - accuracy: 0.9939 - loss:
0.0129 - val_accuracy: 1.0000 - val_loss: 9.4250e-05
Epoch 23/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 110s 14s/step - accuracy: 1.0000 - loss:
0.0030 - val_accuracy: 1.0000 - val_loss: 1.1457e-04
Epoch 24/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 103s 13s/step - accuracy: 1.0000 - loss:
0.0019 - val_accuracy: 1.0000 - val_loss: 1.1573e-04
Epoch 25/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 1.0000 - loss:
8.5508e-04 - val_accuracy: 1.0000 - val_loss: 3.2484e-06
Epoch 26/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 1.0000 - loss:
1.9165e-04 - val_accuracy: 1.0000 - val_loss: 1.0431e-06
Epoch 27/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 1.0000 - loss:
1.5478e-04 - val_accuracy: 1.0000 - val_loss: 2.7939e-06
Epoch 28/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 1.0000 - loss:
1.8734e-04 - val_accuracy: 1.0000 - val_loss: 4.8575e-06
Epoch 29/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 101s 13s/step - accuracy: 1.0000 - loss:
1.0755e-04 - val_accuracy: 1.0000 - val_loss: 3.9524e-06
Epoch 30/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 99s 12s/step - accuracy: 1.0000 - loss:
8.3568e-05 - val_accuracy: 1.0000 - val_loss: 2.3618e-06
```

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

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

```
<Figure size 1000x1000 with 0 Axes>

<Figure size 1000x1000 with 0 Axes>
```

```
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()
```

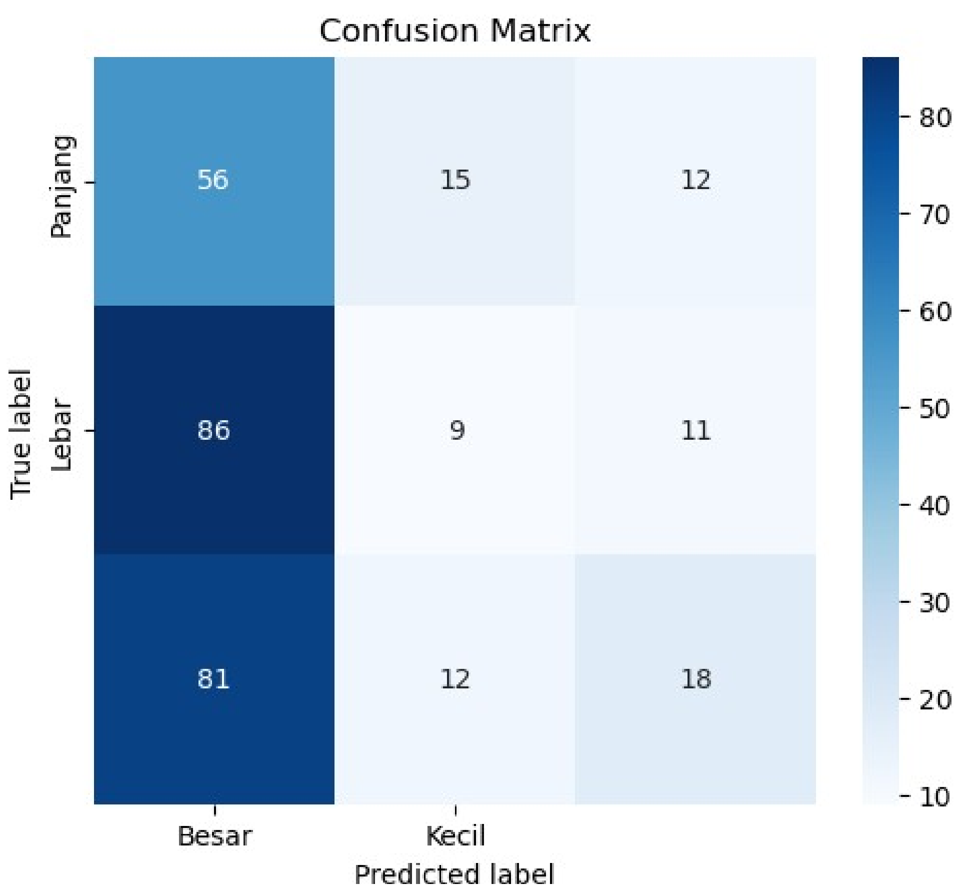



```
import streamlit as st

st.write("Evaluasi Model pada Test Set")
test_loss, test_accuracy = model.evaluate(test_ds)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")

1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 1.0000 - loss:
1.1921e-07
Test Loss: 1.1920925402364446e-07
Test Accuracy: 1.0

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

```
# Load model
model = load_model(r'D:\ATMAJAYA\Semester5\PMDPL\UAS_A_11648\
BestModel_VGG-16_Transformers.h5')

# Load test data
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(224, 224)  # Ubah ukuran ke 224x224 agar sesuai dengan
input model VGG-16
)

# Normalisasi data jika model mengharuskan input dalam rentang [0, 1]
test_data = test_data.map(lambda x, y: (x / 255.0, y))

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

# Ekstrak label sebenarnya 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 akurasi, presisi, recall, dan F1 score
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)
precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=1)
f1_score = 2 * (precision * recall) / (precision + recall)

# Visualisasi Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Besar", "Kecil"], yticklabels=["Panjang",
"Lebar"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Menampilkan hasil
```

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

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

Found 300 files belonging to 3 classes.
10/10 ━━━━━━━━━━━━━━━━━━━━ 35s 3s/step
```




```
Confusion Matrix:
 [[56 15 12]
 [86  9 11]
 [81 12 18]]
Akurasi: 0.27666666666666667
Presisi: [0.25112108 0.25       0.43902439]
Recall: [0.6746988  0.08490566 0.16216216]
F1 Score: [0.36601307 0.12676056 0.23684211]
```

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

# Memuat model yang sudah dilatih
model = load_model(r'D:\ATMAJAYA\Semester5\PMDPL\UAS_A_11648\
BestModel_VGG-16_Transformers.h5')  # Ganti dengan path model Anda
class_names = ['Busuk','Matang', 'Mentah']  # Kelas yang ada pada
model

# Fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        # Memuat dan mempersiapkan gambar untuk prediksi dengan ukuran
yang sesuai
        input_image = tf.keras.utils.load_img(image_path,
target_size=(224, 224))  # Ubah ukuran gambar sesuai dengan model
        input_image_array = tf.keras.utils.img_to_array(input_image)
# Mengubah gambar jadi array numpy agar bisa di proses model
        input_image_array =
tf.keras.applications.vgg16.preprocess_input(input_image_array)  #
Preprocessing gambar

        # Menambahkan dimensi batch agar sesuai dengan input model
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)  #
Dimensi menjadi (1, 224, 224, 3)

        # Melakukan prediksi
        predictions = model.predict(input_image_exp_dim)  # Melakukan
prediksi pada gambar yang telah diproses

        # Cek bentuk predictions
        print(f"Predictions shape: {predictions.shape}")

        # Pastikan predictions memiliki hasil yang benar
        if predictions.shape[0] == 0:
            return "Terjadi kesalahan: Tidak ada hasil prediksi."

        result = tf.nn.softmax(predictions[0])  # Menghitung hasil
prediksi
        class_idx = np.argmax(result)  # Menemukan indeks kelas
        confidence = np.max(result) * 100  # Menghitung confidence
dalam persentase

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

```
confidence

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

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

# Contoh penggunaan fungsi
result = classify_images(r'D:\ATMAJAYA\Semester5\PMDPL\UAS_A_11648\
test_data\Matang\6.jpg', save_path='matang.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 you train
or evaluate the model.
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 388ms/step
Predictions shape: (1, 3)
Prediksi: Mentah
Confidence: 41.43%
Prediksi: Mentah dengan confidence 41.43%. Gambar asli disimpan di
matang.jpg.
```

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

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

Nama : Axel Nicolas Kusmanto Putra NPM : 220711677 Kelompok SB : Transformers Topik : Klasifikasi Kematangan Rambutan Arsitektur : GoogleNet

```
import tensorflow as tf
# import cv2
import numpy as np
from matplotlib import pyplot as plt
#load data
data_dir = r"C:\Users\ACER\Downloads\Dataset\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=(224, 224), batch_size=32)
print(data.class_names)

class_names = data.class_names
```

```
Found 317 files belonging to 3 classes.
['Busuk', 'Matang', 'Mentah']
```

```
img_size = 224
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,
)
```

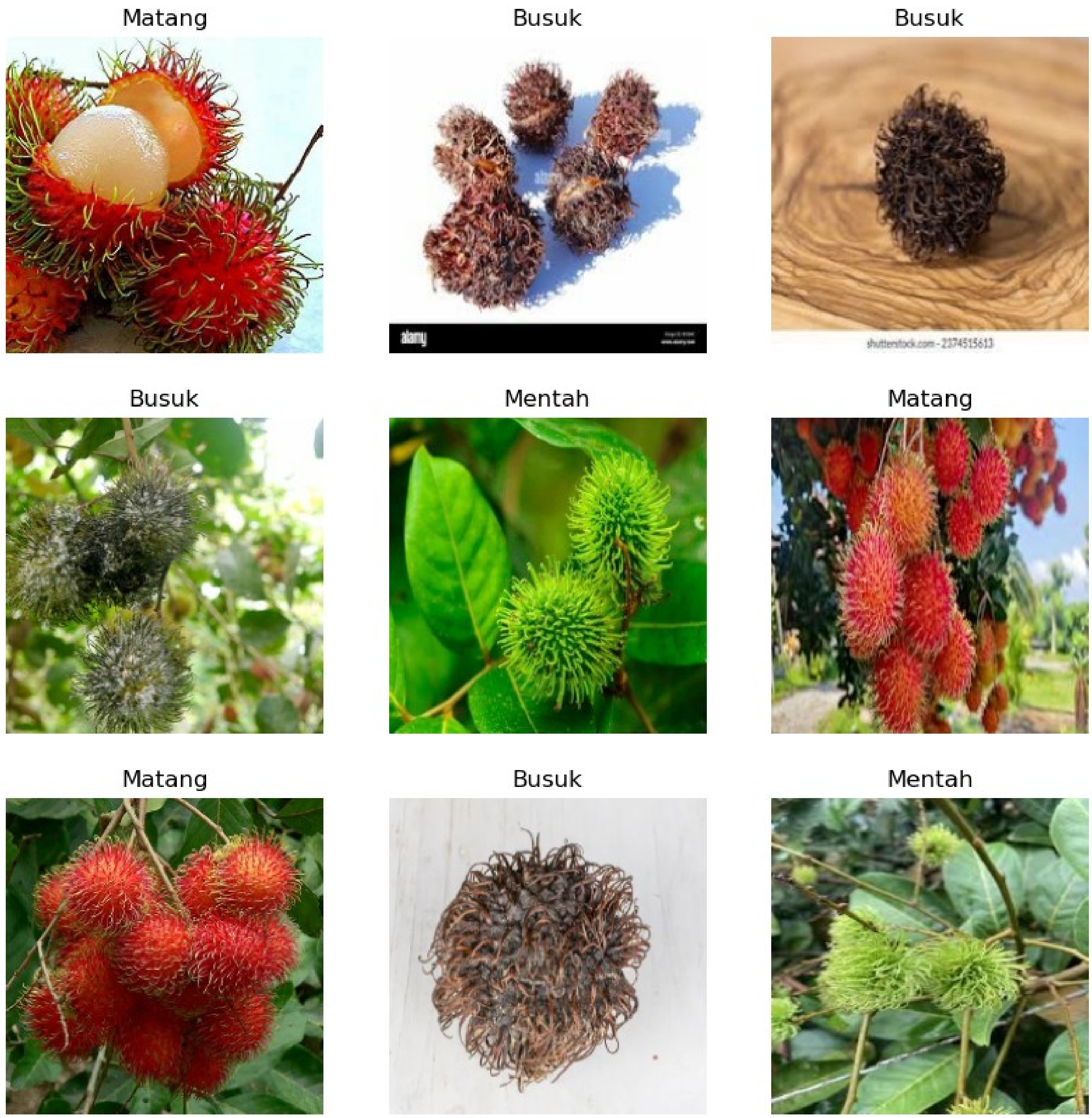
```
Found 317 files belonging to 3 classes.
```

```
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)

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

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

```
import matplotlib.pyplot as plt

i = 0
```

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

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

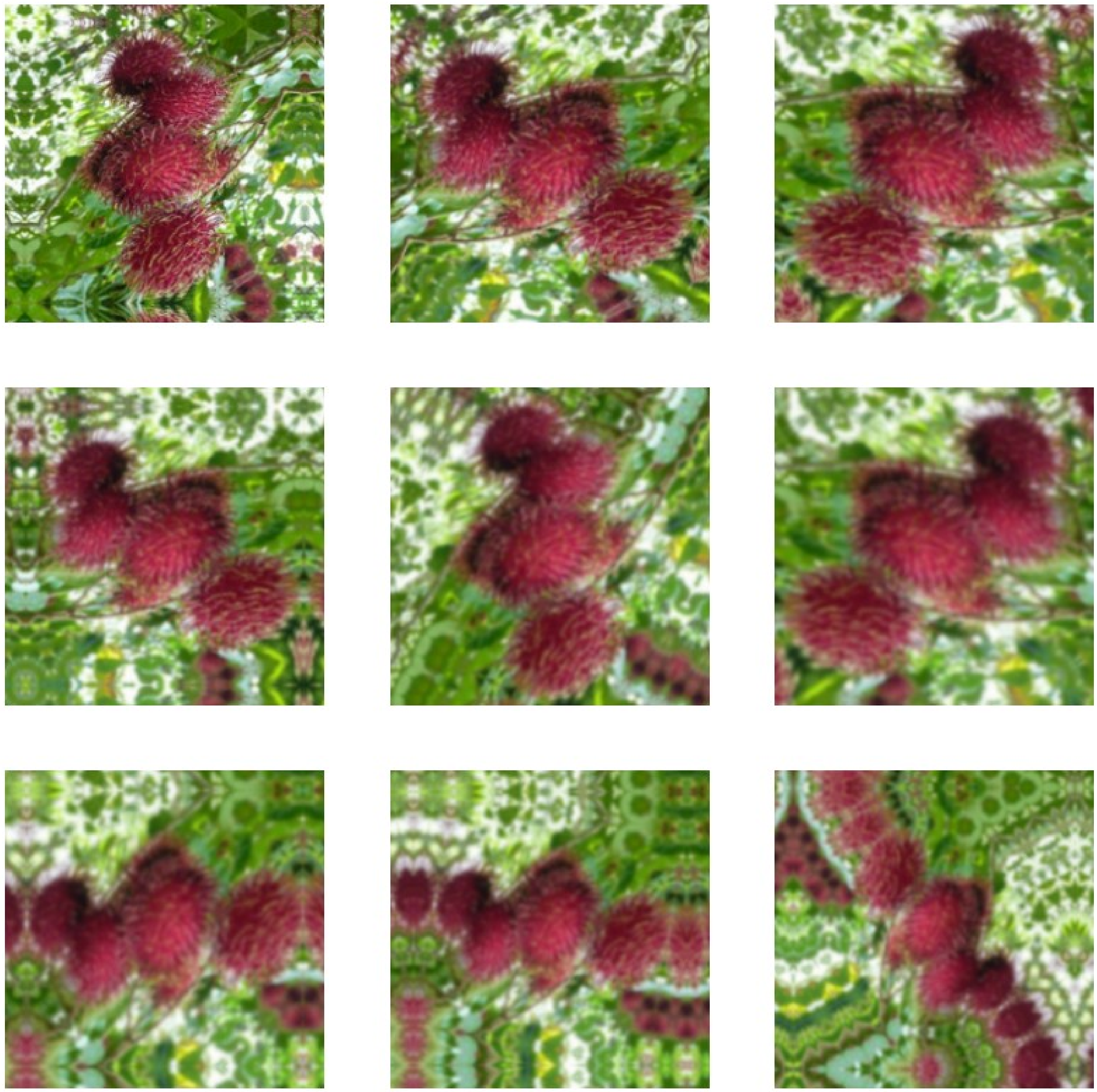


```
for images, labels in data.take(1):
    images_array = np.array(images)
    print(images_array.shape)

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

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

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

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

data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape=(img_size, img_size,
3)),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
])


i = 0
plt.figure(figsize=(10,10))
#Lihat data setelah di augmentasi
for images, labels in data.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\ACER\anaconda3\Lib\site-packages\keras\src\layers\
preprocessing\tf_data_layer.py:19: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```



```
import tensorflow as tf
import keras

import keras._tf_keras.keras.backend as K
from keras._tf_keras.keras.models import Model
from keras._tf_keras.keras.layers import Input, Dense, Conv2D
from keras._tf_keras.keras.layers import Flatten, MaxPool2D, AvgPool2D
from keras._tf_keras.keras.layers import Concatenate, Dropout
from keras._tf_keras.keras.regularizers import l2
from keras._tf_keras.keras.models import load_model

#membuat model from scratch
```

```
def googlenet(input_shape, n_classes):

    def inception_block(x, f):
        t1 = Conv2D(f[0], 1, activation='relu',
kernel_regularizer=l2(0.01))(x)

        t2 = Conv2D(f[1], 1, activation='relu',
kernel_regularizer=l2(0.01))(x)
        t2 = Conv2D(f[2], 3, padding='same', activation='relu',
kernel_regularizer=l2(0.01))(t2)

        t3 = Conv2D(f[3], 1, activation='relu',
kernel_regularizer=l2(0.01))(x)
        t3 = Conv2D(f[4], 5, padding='same', activation='relu',
kernel_regularizer=l2(0.01))(t3)

        t4 = MaxPool2D(3, 1, padding='same')(x)
        t4 = Conv2D(f[5], 1, activation='relu',
kernel_regularizer=l2(0.01))(t4)
        output = Concatenate()([t1, t2, t3, t4])
        return output


    input = Input(input_shape)

    x = Conv2D(64, 7, strides=2, padding='same', activation='relu',
kernel_regularizer=l2(0.01))(input)
    x = MaxPool2D(3, strides=2, padding='same')(x)

    x = Conv2D(64, 1, activation='relu', kernel_regularizer=l2(0.01))
(x)
    x = Conv2D(192, 3, padding='same', activation='relu',
kernel_regularizer=l2(0.01))(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(0.5)(x)

    x = Flatten()(x)
    output = Dense(n_classes, activation='softmax',
kernel_regularizer=l2(0.01))(x)

    model = Model(input, output)
    return model
#Pastikan input shae dan jumlah kelas sesuai
input_shape = 224, 224, 3
n_classes = 3

#Clear Cache Keras menggunakan clear session
K.clear_session()
#buat model dengan
###Terdapat code yang hilang disini! lihat modul untuk menemukanya

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

```
Model: "functional"

┏━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━
━━━━━┓
┃ Layer (type)         ┃ Output Shape       ┃    Param # ┃ Connected to
┃
┡━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━
━━━━━┩
│ input_layer          │ (None, 224, 224,   │          0 │ -
│
│ (InputLayer)         │ 3)                 │            │
│
├──────────────────────┼────────────────────┼────────────┼───────────────────
─────┤
│ conv2d (Conv2D)      │ (None, 112, 112,   │      9,472 │
input_layer[0][0] │
│                      │ 64)                │            │
│
├──────────────────────┼────────────────────┼────────────┼───────────────────
─────┤
│ max_pooling2d        │ (None, 56, 56,     │          0 │ conv2d[0][0]
│
│ (MaxPooling2D)       │ 64)                │            │
│
├──────────────────────┼────────────────────┼────────────┼───────────────────
─────┤
│ conv2d_1 (Conv2D)    │ (None, 56, 56,     │      4,160 │
max_pooling2d[0]… │
│                      │ 64)                │            │
│
```

```
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ conv2d_2 (Conv2D)    │ (None, 56, 56,    │    110,784 │ conv2d_1[0]
[0]     │
│                      │ 192)              │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ max_pooling2d_1      │ (None, 27, 27,    │          0 │ conv2d_2[0]
[0]     │
│ (MaxPooling2D)       │ 192)              │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ conv2d_4 (Conv2D)    │ (None, 27, 27,    │     18,528 │
max_pooling2d_1[… │
│                      │ 96)               │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ conv2d_6 (Conv2D)    │ (None, 27, 27,    │      3,088 │
max_pooling2d_1[… │
│                      │ 16)               │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ max_pooling2d_2      │ (None, 27, 27,    │          0 │
max_pooling2d_1[… │
│ (MaxPooling2D)       │ 192)              │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ conv2d_3 (Conv2D)    │ (None, 27, 27,    │     12,352 │
max_pooling2d_1[… │
│                      │ 64)               │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ conv2d_5 (Conv2D)    │ (None, 27, 27,    │    110,720 │ conv2d_4[0]
[0]     │
│                      │ 128)              │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
───────┤
│ conv2d_7 (Conv2D)    │ (None, 27, 27,    │     12,832 │ conv2d_6[0]
[0]     │
│                      │ 32)               │            │
│
├──────────────────────┼───────────────────┼────────────┼──────────────
```

```
───────┤
│ conv2d_8 (Conv2D)    │ (None, 27, 27,      │      6,176 │
max_pooling2d_2[… │
│                      │ 32)                 │            │
│
├──────────────────────┼─────────────────────┼────────────┼──────────────
───────┤
│ concatenate          │ (None, 27, 27,      │          0 │ conv2d_3[0]
[0],    │
│ (Concatenate)        │ 256)                │            │ conv2d_5[0]
[0],    │
│                      │                     │            │ conv2d_7[0]
[0],    │
│                      │                     │            │ conv2d_8[0]
[0]     │
├──────────────────────┼─────────────────────┼────────────┼──────────────
───────┤
│ conv2d_10 (Conv2D)   │ (None, 27, 27,      │     32,896 │
concatenate[0][0] │
│                      │ 128)                │            │
│
├──────────────────────┼─────────────────────┼────────────┼──────────────
───────┤
│ conv2d_12 (Conv2D)   │ (None, 27, 27,      │      8,224 │
concatenate[0][0] │
│                      │ 32)                 │            │
│
├──────────────────────┼─────────────────────┼────────────┼──────────────
───────┤
│ max_pooling2d_3      │ (None, 27, 27,      │          0 │
concatenate[0][0] │
│ (MaxPooling2D)       │ 256)                │            │
│
├──────────────────────┼─────────────────────┼────────────┼──────────────
───────┤
│ conv2d_9 (Conv2D)    │ (None, 27, 27,      │     32,896 │
concatenate[0][0] │
│                      │ 128)                │            │
│
├──────────────────────┼─────────────────────┼────────────┼──────────────
───────┤
│ conv2d_11 (Conv2D)   │ (None, 27, 27,      │    221,376 │ conv2d_10[0]
[0]    │
│                      │ 192)                │            │
│
├──────────────────────┼─────────────────────┼────────────┼──────────────
───────┤
│ conv2d_13 (Conv2D)   │ (None, 27, 27,      │     76,896 │ conv2d_12[0]
[0]    │
```

```
│                          │ 96)                    │               │
│
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
──────┤
│ conv2d_14 (Conv2D)       │ (None, 27, 27,         │        16,448 │
max_pooling2d_3[… │
│                          │ 64)                    │               │
│
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
──────┤
│ concatenate_1            │ (None, 27, 27,         │             0 │ conv2d_9[0]
[0],    │
│ (Concatenate)            │ 480)                   │               │ conv2d_11[0]
[0],   │
│                          │                        │               │ conv2d_13[0]
[0],   │
│                          │                        │               │ conv2d_14[0]
[0]    │
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
──────┤
│ max_pooling2d_4          │ (None, 14, 14,         │             0 │
concatenate_1[0]… │
│ (MaxPooling2D)           │ 480)                   │               │
│
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
──────┤
│ conv2d_16 (Conv2D)       │ (None, 14, 14,         │        46,176 │
max_pooling2d_4[… │
│                          │ 96)                    │               │
│
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
──────┤
│ conv2d_18 (Conv2D)       │ (None, 14, 14,         │         7,696 │
max_pooling2d_4[… │
│                          │ 16)                    │               │
│
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
──────┤
│ max_pooling2d_5          │ (None, 14, 14,         │             0 │
max_pooling2d_4[… │
│ (MaxPooling2D)           │ 480)                   │               │
│
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
──────┤
│ conv2d_15 (Conv2D)       │ (None, 14, 14,         │        92,352 │
max_pooling2d_4[… │
│                          │ 192)                   │               │
│
├──────────────────────────┼────────────────────────┼───────────────┼───────────────
```

```
──────┤
│ conv2d_17 (Conv2D)  │ (None, 14, 14,      │    179,920 │ conv2d_16[0]
[0]    │
│                     │ 208)                │            │
│
├─────────────────────┼─────────────────────┼────────────┼─────────────
──────┤
│ conv2d_19 (Conv2D)  │ (None, 14, 14,      │     19,248 │ conv2d_18[0]
[0]    │
│                     │ 48)                 │            │
│
├─────────────────────┼─────────────────────┼────────────┼─────────────
──────┤
│ conv2d_20 (Conv2D)  │ (None, 14, 14,      │     30,784 │
max_pooling2d_5[… │
│                     │ 64)                 │            │
│
├─────────────────────┼─────────────────────┼────────────┼─────────────
──────┤
│ concatenate_2       │ (None, 14, 14,      │          0 │ conv2d_15[0]
[0],   │
│ (Concatenate)       │ 512)                │            │ conv2d_17[0]
[0],   │
│                     │                     │            │ conv2d_19[0]
[0],   │
│                     │                     │            │ conv2d_20[0]
[0]    │
├─────────────────────┼─────────────────────┼────────────┼─────────────
──────┤
│ conv2d_22 (Conv2D)  │ (None, 14, 14,      │     57,456 │
concatenate_2[0]… │
│                     │ 112)                │            │
│
├─────────────────────┼─────────────────────┼────────────┼─────────────
──────┤
│ conv2d_24 (Conv2D)  │ (None, 14, 14,      │     12,312 │
concatenate_2[0]… │
│                     │ 24)                 │            │
│
├─────────────────────┼─────────────────────┼────────────┼─────────────
──────┤
│ max_pooling2d_6     │ (None, 14, 14,      │          0 │
concatenate_2[0]… │
│ (MaxPooling2D)      │ 512)                │            │
│
├─────────────────────┼─────────────────────┼────────────┼─────────────
──────┤
│ conv2d_21 (Conv2D)  │ (None, 14, 14,      │     82,080 │
concatenate_2[0]… │
```

```
│                         │ 160)                   │               │
│
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
──────┤
│ conv2d_23 (Conv2D)      │ (None, 14, 14,         │       226,016 │ conv2d_22[0]
[0]    │
│                         │ 224)                   │               │
│
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
──────┤
│ conv2d_25 (Conv2D)      │ (None, 14, 14,         │        38,464 │ conv2d_24[0]
[0]    │
│                         │ 64)                    │               │
│
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
──────┤
│ conv2d_26 (Conv2D)      │ (None, 14, 14,         │        32,832 │
max_pooling2d_6[… │
│                         │ 64)                    │               │
│
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
──────┤
│ concatenate_3           │ (None, 14, 14,         │             0 │ conv2d_21[0]
[0],   │
│ (Concatenate)           │ 512)                   │               │ conv2d_23[0]
[0],   │
│                         │                        │               │ conv2d_25[0]
[0],   │
│                         │                        │               │ conv2d_26[0]
[0]    │
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
──────┤
│ conv2d_28 (Conv2D)      │ (None, 14, 14,         │        65,664 │
concatenate_3[0]… │
│                         │ 128)                   │               │
│
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
──────┤
│ conv2d_30 (Conv2D)      │ (None, 14, 14,         │        12,312 │
concatenate_3[0]… │
│                         │ 24)                    │               │
│
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
──────┤
│ max_pooling2d_7         │ (None, 14, 14,         │             0 │
concatenate_3[0]… │
│ (MaxPooling2D)          │ 512)                   │               │
│
├─────────────────────────┼────────────────────────┼───────────────┼─────────────────
```

```
──────┤
│ conv2d_27 (Conv2D)  │ (None, 14, 14,      │     65,664 │
concatenate_3[0]… │
│                     │ 128)                │            │
│
├─────────────────────┼─────────────────────┼────────────┼──────────
──────┤
│ conv2d_29 (Conv2D)  │ (None, 14, 14,      │    295,168 │ conv2d_28[0]
[0]     │
│                     │ 256)                │            │
│
├─────────────────────┼─────────────────────┼────────────┼──────────
──────┤
│ conv2d_31 (Conv2D)  │ (None, 14, 14,      │     38,464 │ conv2d_30[0]
[0]     │
│                     │ 64)                 │            │
│
├─────────────────────┼─────────────────────┼────────────┼──────────
──────┤
│ conv2d_32 (Conv2D)  │ (None, 14, 14,      │     32,832 │
max_pooling2d_7[… │
│                     │ 64)                 │            │
│
├─────────────────────┼─────────────────────┼────────────┼──────────
──────┤
│ concatenate_4       │ (None, 14, 14,      │          0 │ conv2d_27[0]
[0],    │
│ (Concatenate)       │ 512)                │            │ conv2d_29[0]
[0],    │
│                     │                     │            │ conv2d_31[0]
[0],    │
│                     │                     │            │ conv2d_32[0]
[0]     │
├─────────────────────┼─────────────────────┼────────────┼──────────
──────┤
│ conv2d_34 (Conv2D)  │ (None, 14, 14,      │     73,872 │
concatenate_4[0]… │
│                     │ 144)                │            │
│
├─────────────────────┼─────────────────────┼────────────┼──────────
──────┤
│ conv2d_36 (Conv2D)  │ (None, 14, 14,      │     16,416 │
concatenate_4[0]… │
│                     │ 32)                 │            │
│
├─────────────────────┼─────────────────────┼────────────┼──────────
──────┤
│ max_pooling2d_8     │ (None, 14, 14,      │          0 │
concatenate_4[0]… │
```

```
│ (MaxPooling2D)        │ 512)              │              │
│
├───────────────────────┼───────────────────┼──────────────┼──────────────
──────┤
│ conv2d_33 (Conv2D)    │ (None, 14, 14,    │       57,456 │
concatenate_4[0]… │
│                       │ 112)              │              │
│
├───────────────────────┼───────────────────┼──────────────┼──────────────
──────┤
│ conv2d_35 (Conv2D)    │ (None, 14, 14,    │      373,536 │ conv2d_34[0]
[0]     │
│                       │ 288)              │              │
│
├───────────────────────┼───────────────────┼──────────────┼──────────────
──────┤
│ conv2d_37 (Conv2D)    │ (None, 14, 14,    │       51,264 │ conv2d_36[0]
[0]     │
│                       │ 64)               │              │
│
├───────────────────────┼───────────────────┼──────────────┼──────────────
──────┤
│ conv2d_38 (Conv2D)    │ (None, 14, 14,    │       32,832 │
max_pooling2d_8[… │
│                       │ 64)               │              │
│
├───────────────────────┼───────────────────┼──────────────┼──────────────
──────┤
│ concatenate_5         │ (None, 14, 14,    │            0 │ conv2d_33[0]
[0],    │
│ (Concatenate)         │ 528)              │              │ conv2d_35[0]
[0],    │
│                       │                   │              │ conv2d_37[0]
[0],    │
│                       │                   │              │ conv2d_38[0]
[0]     │
├───────────────────────┼───────────────────┼──────────────┼──────────────
──────┤
│ conv2d_40 (Conv2D)    │ (None, 14, 14,    │       84,640 │
concatenate_5[0]… │
│                       │ 160)              │              │
│
├───────────────────────┼───────────────────┼──────────────┼──────────────
──────┤
│ conv2d_42 (Conv2D)    │ (None, 14, 14,    │       16,928 │
concatenate_5[0]… │
│                       │ 32)               │              │
│
├───────────────────────┼───────────────────┼──────────────┼──────────────
```

```
───────┤
│ max_pooling2d_9      │ (None, 14, 14,     │          0 │
concatenate_5[0]… │
│ (MaxPooling2D)       │ 528)               │            │
│
├──────────────────────┼────────────────────┼────────────┼──────────────
───────┤
│ conv2d_39 (Conv2D)   │ (None, 14, 14,     │    135,424 │
concatenate_5[0]… │
│                      │ 256)               │            │
│
├──────────────────────┼────────────────────┼────────────┼──────────────
───────┤
│ conv2d_41 (Conv2D)   │ (None, 14, 14,     │    461,120 │ conv2d_40[0]
[0]    │
│                      │ 320)               │            │
│
├──────────────────────┼────────────────────┼────────────┼──────────────
───────┤
│ conv2d_43 (Conv2D)   │ (None, 14, 14,     │    102,528 │ conv2d_42[0]
[0]    │
│                      │ 128)               │            │
│
├──────────────────────┼────────────────────┼────────────┼──────────────
───────┤
│ conv2d_44 (Conv2D)   │ (None, 14, 14,     │     67,712 │
max_pooling2d_9[… │
│                      │ 128)               │            │
│
├──────────────────────┼────────────────────┼────────────┼──────────────
───────┤
│ concatenate_6        │ (None, 14, 14,     │          0 │ conv2d_39[0]
[0],   │
│ (Concatenate)        │ 832)               │            │ conv2d_41[0]
[0],   │
│                      │                    │            │ conv2d_43[0]
[0],   │
│                      │                    │            │ conv2d_44[0]
[0]    │
├──────────────────────┼────────────────────┼────────────┼──────────────
───────┤
│ max_pooling2d_10     │ (None, 7, 7, 832)  │          0 │
concatenate_6[0]… │
│ (MaxPooling2D)       │                    │            │
│
├──────────────────────┼────────────────────┼────────────┼──────────────
───────┤
│ conv2d_46 (Conv2D)   │ (None, 7, 7, 160)  │    133,280 │
max_pooling2d_10… │
```

```
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ conv2d_48 (Conv2D)    │ (None, 7, 7, 32)   │      26,656 │
max_pooling2d_10… │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ max_pooling2d_11      │ (None, 7, 7, 832)  │           0 │
max_pooling2d_10… │
│ (MaxPooling2D)        │                    │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ conv2d_45 (Conv2D)    │ (None, 7, 7, 256)  │     213,248 │
max_pooling2d_10… │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ conv2d_47 (Conv2D)    │ (None, 7, 7, 320)  │     461,120 │ conv2d_46[0]
[0]    │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ conv2d_49 (Conv2D)    │ (None, 7, 7, 128)  │     102,528 │ conv2d_48[0]
[0]    │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ conv2d_50 (Conv2D)    │ (None, 7, 7, 128)  │     106,624 │
max_pooling2d_11… │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ concatenate_7         │ (None, 7, 7, 832)  │           0 │ conv2d_45[0]
[0],   │
│ (Concatenate)         │                    │             │ conv2d_47[0]
[0],   │
│                       │                    │             │ conv2d_49[0]
[0],   │
│                       │                    │             │ conv2d_50[0]
[0]    │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ conv2d_52 (Conv2D)    │ (None, 7, 7, 192)  │     159,936 │
concatenate_7[0]… │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ conv2d_54 (Conv2D)    │ (None, 7, 7, 48)   │      39,984 │
concatenate_7[0]… │
├───────────────────────┼────────────────────┼─────────────┼──────────────────
─────┤
│ max_pooling2d_12      │ (None, 7, 7, 832)  │           0 │
concatenate_7[0]… │
│ (MaxPooling2D)        │                    │             │
```

```
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_51 (Conv2D)    │ (None, 7, 7, 384)  │     319,872 │
concatenate_7[0]… │
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_53 (Conv2D)    │ (None, 7, 7, 384)  │     663,936 │ conv2d_52[0]
[0]    │
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_55 (Conv2D)    │ (None, 7, 7, 128)  │     153,728 │ conv2d_54[0]
[0]    │
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_56 (Conv2D)    │ (None, 7, 7, 128)  │     106,624 │
max_pooling2d_12… │
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ concatenate_8         │ (None, 7, 7,       │           0 │ conv2d_51[0]
[0],   │
│ (Concatenate)         │ 1024)              │             │ conv2d_53[0]
[0],   │
│                       │                    │             │ conv2d_55[0]
[0],   │
│                       │                    │             │ conv2d_56[0]
[0]    │
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ average_pooling2d     │ (None, 5, 5,       │           0 │
concatenate_8[0]… │
│ (AveragePooling2D)    │ 1024)              │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ dropout (Dropout)     │ (None, 5, 5,       │           0 │
average_pooling2… │
│                       │ 1024)              │             │
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ flatten (Flatten)     │ (None, 25600)      │           0 │ dropout[0][0]
│
├───────────────────────┼────────────────────┼─────────────┼──────────────
───────┤
│ dense (Dense)         │ (None, 3)          │      76,803 │ flatten[0][0]
│
└───────────────────────┴────────────────────┴─────────────┴──────────────
───────┘
```

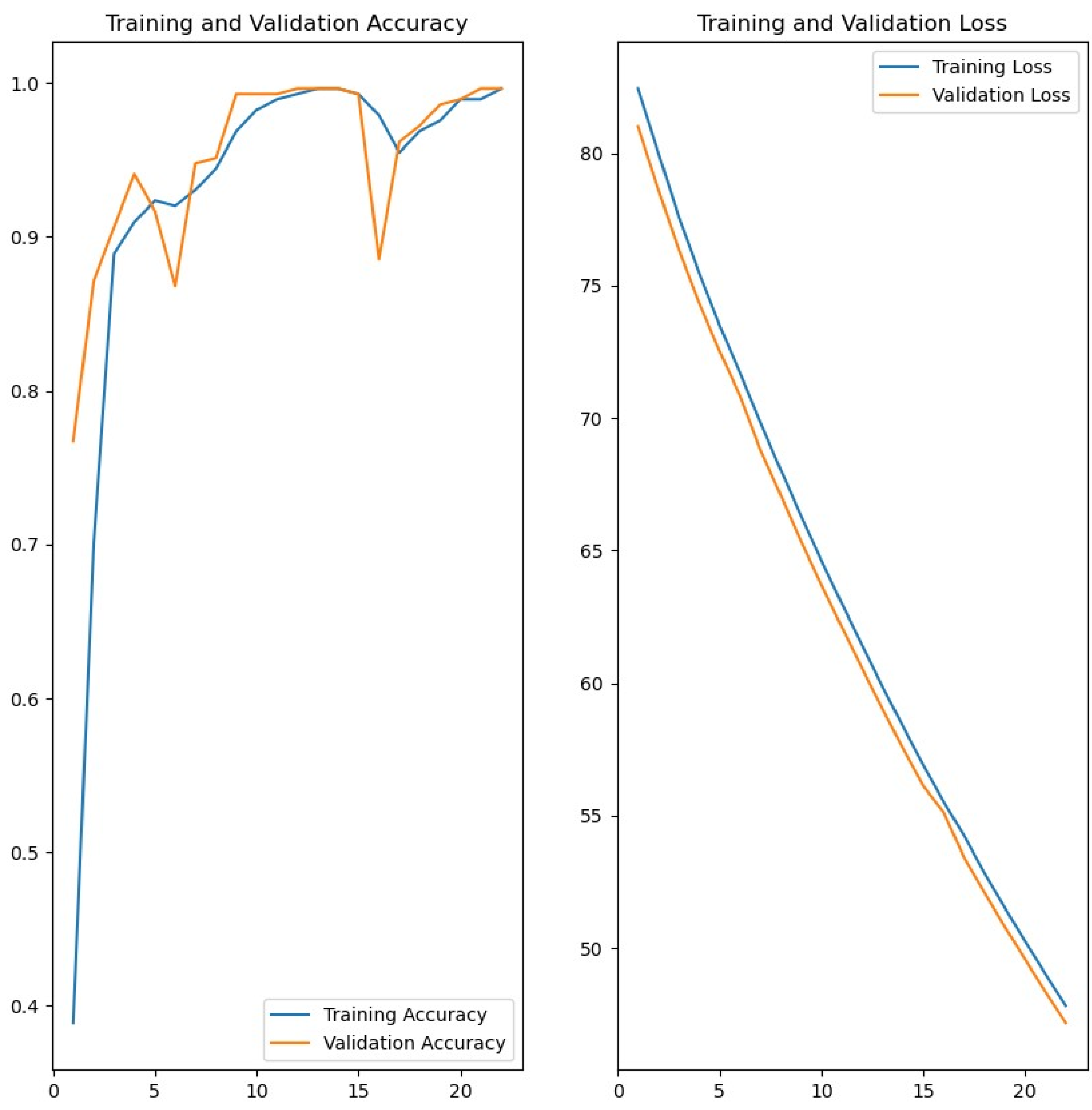
```
 Total params: 6,050,355 (23.08 MB)

 Trainable params: 6,050,355 (23.08 MB)

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

```python
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
#Coimpile dengan optimizer adam
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
model.compile(
    optimizer=Adam(learning_rate=1e-4),  # Sesuaikan learning rate
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

#buat early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=10,
                               mode='max')
#fit validation data ke dalam model
history= model.fit(data,
                   epochs=30,
                   validation_data=val_ds,
                   callbacks=[early_stopping])
```

```
Epoch 1/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 46s 2s/step - accuracy: 0.3819 - loss:
82.9665 - val_accuracy: 0.7674 - val_loss: 81.0007
Epoch 2/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.6478 - loss:
80.4822 - val_accuracy: 0.8715 - val_loss: 78.6140
Epoch 3/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.8825 - loss:
78.0505 - val_accuracy: 0.9062 - val_loss: 76.3505
Epoch 4/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9250 - loss:
75.8457 - val_accuracy: 0.9410 - val_loss: 74.3389
Epoch 5/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9115 - loss:
73.8900 - val_accuracy: 0.9167 - val_loss: 72.5330
Epoch 6/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9220 - loss:
72.0702 - val_accuracy: 0.8681 - val_loss: 70.8434
Epoch 7/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9295 - loss:
70.2006 - val_accuracy: 0.9479 - val_loss: 68.8061
Epoch 8/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9399 - loss:
68.3882 - val_accuracy: 0.9514 - val_loss: 67.0882
```

```
Epoch 9/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9555 - loss:
66.6650 - val_accuracy: 0.9931 - val_loss: 65.3482
Epoch 10/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 20s 2s/step - accuracy: 0.9695 - loss:
64.9862 - val_accuracy: 0.9931 - val_loss: 63.7076
Epoch 11/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9813 - loss:
63.3371 - val_accuracy: 0.9931 - val_loss: 62.1146
Epoch 12/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 20s 2s/step - accuracy: 0.9945 - loss:
61.7313 - val_accuracy: 0.9965 - val_loss: 60.5472
Epoch 13/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9955 - loss:
60.1782 - val_accuracy: 0.9965 - val_loss: 59.0228
Epoch 14/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9993 - loss:
58.6517 - val_accuracy: 0.9965 - val_loss: 57.5397
Epoch 15/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9913 - loss:
57.1963 - val_accuracy: 0.9931 - val_loss: 56.1167
Epoch 16/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9857 - loss:
55.7851 - val_accuracy: 0.8854 - val_loss: 55.1253
Epoch 17/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9394 - loss:
54.5513 - val_accuracy: 0.9618 - val_loss: 53.4137
Epoch 18/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9661 - loss:
53.0878 - val_accuracy: 0.9722 - val_loss: 52.0988
Epoch 19/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.9587 - loss:
51.7999 - val_accuracy: 0.9861 - val_loss: 50.8145
Epoch 20/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 19s 2s/step - accuracy: 0.9913 - loss:
50.4919 - val_accuracy: 0.9896 - val_loss: 49.5652
Epoch 21/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 23s 3s/step - accuracy: 0.9894 - loss:
49.2519 - val_accuracy: 0.9965 - val_loss: 48.3459
Epoch 22/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 23s 3s/step - accuracy: 0.9965 - loss:
48.0553 - val_accuracy: 0.9965 - val_loss: 47.1704

#buat plot dengan menggunakan history supaya jumlahnya sesuai epoch
yang dilakukan
ephocs_range = range(1, len(history.history['loss']) + 1)
plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.plot(ephocs_range, history.history['accuracy'], label='Training
Accuracy')
```

```
plt.plot(ephocs_range, history.history['val_accuracy'],
label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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



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

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

```
import 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'C:\Users\ACER\Downloads\UAS ML\gugelnet.h5')  #
Ganti dengan path model Anda
class_names = ['Busuk','Matang', 'Mentah']

# 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=(224, 224))
        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
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
```

```
result = classify_images(r'C:\Users\ACER\Downloads\Dataset\test_data\
Mentah\1.jpg', save_path='mentah.jpg')
print(result)
```

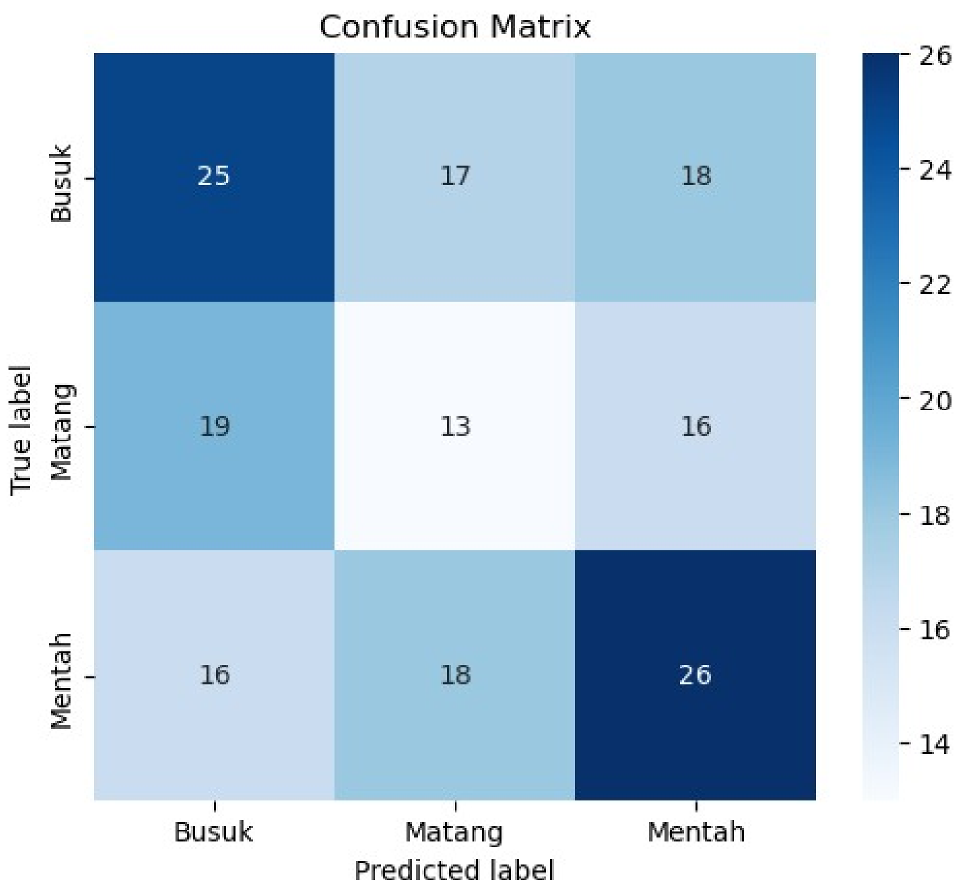
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 888ms/step
Prediksi: Mentah
Confidence: 57.39%
Prediksi: Mentah dengan confidence 57.39%. Gambar asli disimpan di
mentah.jpg.
```

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

# Muat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\ACER\Downloads\Dataset\test_data',
    labels='inferred',
    label_mode='categorical',  # Menghasilkan label dalam bentuk one-
hot encodin
    batch_size=64,
    image_size=(224, 224)
)

# Prediksi model
y_pred = 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 = []
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)

# Membuat matriks kebingungan
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)


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

# Menghitung presisi dan recall
```

```
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))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Busuk","Matang","Mentah"],
yticklabels=["Busuk","Matang","Mentah"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
# Menampilkan hasil
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
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 168 files belonging to 3 classes.
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 1s/step
```




```
Confusion Matrix:
 [[25 17 18]
 [19 13 16]
 [16 18 26]]
Akurasi: 0.38095238095238093
Presisi: [0.41666667 0.27083333 0.43333333]
Recall: [0.41666667 0.27083333 0.43333333]
F1 Score: [0.41666667 0.27083333 0.43333333]
```