Name: Angelle Grace Dimaguiba

NPM: 2107051

Name: Archipora Nur Qudsia

NPM: 2107041

Kelompok CB : Transformer

Anggota : Gorgeous

```
# Import TensorFlow 2.x
%tensorflow_version 2.x
from google.colab import drive
drive.mount('/content/drive')
data_dir = '/content/drive/MyDrive/Kuliah/Semester 4/Pengolahan Citra/Tugas Besar/Dataset/Dataset_Fruits_Vegetables_Training_Validation_Test'
import os, shutil, glob
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.callbacks import ModelCheckpoint
import numpy as np
import matplotlib.pyplot as plt

img_size = 227
batch_size = 32
train_gen = ImageDataGenerator(rescale=1./255)
train_data = train_gen.flow_from_directory(data_dir, target=(227, 227), classes=['Banana', 'Carrot', 'Cucumber', 'Mixed_Vegetables', 'Mixed_Fruits'], batch_size=32)

train_samples = len(train_data.filenames)
num_classes = len(train_data.class_indices)
print('Found %d images.' % train_samples)
print('Found %d classes.' % num_classes)
print('Class indices: ', train_data.class_indices)
print('Class labels: ', sorted(train_data.class_indices.keys()))

# Ambil batch pertama dan tampilkan
x, y = next(train_data)
print('Shape of x: ', x.shape)
print('Shape of y: ', y.shape)

# Tampilkan 9 gambar
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x[i])
    plt.title('Class: %d' % y[i])
    plt.axis('off')
plt.show()
```

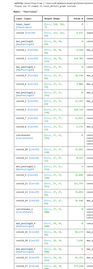
```
Drive already mounted; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Found 15000 images.
Found 5 classes.
Class indices:  {'Banana': 0, 'Carrot': 1, 'Cucumber': 2, 'Mixed_Vegetables': 3, 'Mixed_Fruits': 4}
Class labels:  ['Banana', 'Carrot', 'Cucumber', 'Mixed_Vegetables', 'Mixed_Fruits']
Shape of x:  (32, 227, 227, 3)
Shape of y:  (32, 5)
```

```
# Import TensorFlow 2.x
%tensorflow_version 2.x
from google.colab import drive
drive.mount('/content/drive')
data_dir = '/content/drive/MyDrive/Kuliah/Semester 4/Pengolahan Citra/Tugas Besar/Dataset/Dataset_Fruits_Vegetables_Training_Validation_Test'
import os, shutil, glob
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.callbacks import ModelCheckpoint
import numpy as np
import matplotlib.pyplot as plt

# Generator untuk validasi
val_gen = ImageDataGenerator(rescale=1./255)
val_data = val_gen.flow_from_directory(data_dir, target=(227, 227), classes=['Banana', 'Carrot', 'Cucumber', 'Mixed_Vegetables', 'Mixed_Fruits'], batch_size=32)

val_samples = len(val_data.filenames)
print('Found %d images.' % val_samples)

# Ambil batch pertama dan tampilkan
x, y = next(val_data)
print('Shape of x: ', x.shape)
print('Shape of y: ', y.shape)

# Tampilkan 9 gambar
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x[i])
    plt.title('Class: %d' % y[i])
    plt.axis('off')
plt.show()
```

```
Drive already mounted; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Found 3000 images.
Shape of x:  (32, 227, 227, 3)
Shape of y:  (32, 5)
```

```
# Generator untuk test
test_gen = ImageDataGenerator(rescale=1./255)
test_data = test_gen.flow_from_directory(data_dir, target=(227, 227), classes=['Banana', 'Carrot', 'Cucumber', 'Mixed_Vegetables', 'Mixed_Fruits'], batch_size=32)

test_samples = len(test_data.filenames)
print('Found %d images.' % test_samples)

# Ambil batch pertama dan tampilkan
x, y = next(test_data)
print('Shape of x: ', x.shape)
print('Shape of y: ', y.shape)

# Tampilkan 9 gambar
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x[i])
    plt.title('Class: %d' % y[i])
    plt.axis('off')
plt.show
```

```
Drive already mounted; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Found 3000 images.
Shape of x:  (32, 227, 227, 3)
Shape of y:  (32, 5)
```




```
# Import TensorFlow 2.x
%tensorflow_version 2.x
from google.colab import drive
drive.mount('/content/drive')
data_dir = '/content/drive/MyDrive/Kuliah/Semester 4/Pengolahan Citra/Tugas Besar/Dataset/Dataset_Fruits_Vegetables_Training_Validation_Test'
import os, shutil, glob
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.callbacks import ModelCheckpoint
import numpy as np
import matplotlib.pyplot as plt

# Model CNN
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(227, 227, 3)))
model.add(MaxPooling2D(2, 2))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dense(5, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['acc'])
model.summary()
```

```
Drive already mounted; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/initializers/initializers_v2.py:3445: UserWarning: `tf.nn.dropout` is deprecated. For rescaling `1 - rate` to `keep_prob`, use `keep_prob=1 - rate`.
Instructions for updated: Please use `rate` instead of `keep_prob`.
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 225, 225, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D  (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 110, 110, 64)      18,496
_________________________________________________________________
max_pooling2d_1 (MaxPooling  (None, 55, 55, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 53, 53, 128)       73,728
_________________________________________________________________
max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       147,456
_________________________________________________________________
max_pooling2d_3 (MaxPooling  (None, 12, 12, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0
_________________________________________________________________
dropout (Dropout)            (None, 18432)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9,437,696
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 2,565
=================================================================
Total params: 9,680,837
Trainable params: 9,680,837
Non-trainable params: 0
_________________________________________________________________
```

```
# Training model
history = model.fit_generator(train_data,
                              steps_per_epoch=train_samples / 32,
                              epochs=50,
                              validation_data=val_data,
                              validation_steps=val_samples / 32)
model.save('/content/drive/MyDrive/Kuliah/Semester 4/Pengolahan Citra/Tugas Besar/Dataset/Dataset_Fruits_Vegetables_Training_Validation_Test/model.h5')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/preprocessing/image.py:104: UserWarning: `Keras 2.0.6 introduced a new
image preprocessing pipeline; one may prefer to stay to the old version for now.
If anything breaks, please file a GitHub issue.
Epoch 1/50
468/468 [==============================] - 1,748s 3s/step - loss: 1.7180 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 2/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 3/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 4/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 5/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 6/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 7/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 8/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 9/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 10/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 11/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 12/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 13/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 14/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 15/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 16/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 17/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 18/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 19/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 20/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 21/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 22/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 23/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 24/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 25/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 26/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 27/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 28/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 29/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 30/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 31/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 32/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 33/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 34/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 35/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 36/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 37/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 38/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 39/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 40/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 41/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 42/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 43/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 44/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 45/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 46/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 47/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 48/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 49/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
Epoch 50/50
468/468 [==============================] - 1,745s 3s/step - loss: 1.6094 - acc: 0.2000 - val_loss: 1.6094 - val_acc: 0.2000
```

```
# Plot training dan validasi
import matplotlib.pyplot as plt
plt.plot(history.history['acc'], label='Training Accuracy')
plt.plot(history.history['val_acc'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.show()

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

```
# Import TensorFlow 2.x
%tensorflow_version 2.x
from google.colab import drive
drive.mount('/content/drive')
data_dir = '/content/drive/MyDrive/Kuliah/Semester 4/Pengolahan Citra/Tugas Besar/Dataset/Dataset_Fruits_Vegetables_Training_Validation_Test'
import os, shutil, glob
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.callbacks import ModelCheckpoint
import numpy as np
import matplotlib.pyplot as plt

# Generator untuk test
test_gen = ImageDataGenerator(rescale=1./255)
test_data = test_gen.flow_from_directory(data_dir, target=(227, 227), classes=['Banana', 'Carrot', 'Cucumber', 'Mixed_Vegetables', 'Mixed_Fruits'], batch_size=32)

test_samples = len(test_data.filenames)
print('Found %d images.' % test_samples)

# Ambil batch pertama dan tampilkan
x, y = next(test_data)
print('Shape of x: ', x.shape)
print('Shape of y: ', y.shape)

# Tampilkan 9 gambar
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x[i])
    plt.title('Class: %d' % y[i])
    plt.axis('off')
plt.show
```

```
Drive already mounted; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Found 3000 images.
Shape of x:  (32, 227, 227, 3)
Shape of y:  (32, 5)
```

```
# Evaluasi model
test_loss, test_acc = model.evaluate_generator(test_data, steps=test_samples / 32)
print('test acc: %f', % test_acc)
```

```
test acc: 0.200000
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, y_pred)
print('Confusion Matrix:')
print(cm)

# Plot confusion matrix
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title('Confusion Matrix')
plt.show()
```

```
Confusion Matrix:
[[600   0   0   0   0]
 [600   0   0   0   0]
 [600   0   0   0   0]
 [600   0   0   0   0]
 [600   0   0   0   0]]
Precision: 0.040000 Recall: 0.200000 F1-Score: 0.066667
Classification Report:
              precision    recall  f1-score   support

      Banana       0.20      1.00      0.33       600
      Carrot       0.00      0.00      0.00       600
    Cucumber       0.00      0.00      0.00       600
Mixed_Veg       0.00      0.00      0.00       600
Mixed_Fruit       0.00      0.00      0.00       600

    accuracy                           0.20      3000
   macro avg       0.04      0.20      0.07      3000
weighted avg       0.04      0.20      0.07      3000
```

Nama : Angello Khara Sitanggang

NPM : 220711833

Nama : Emanuel Enrico Anindya Wibawa

NPM : 220711890

Kelompok SB : Transformer

Arsitektur : MobileNet

In [2]:
```
#Install tensorflow packag

#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, Flatte
```

In [3]:
```
count = 0
dirs = os.listdir(r'train_data')
print('List of Classes: ', dirs)
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')
```

```
List of Classes:  ['Busuk', 'Matang', 'Mentah']
Busuk Folder has 360 Images
Matang Folder has 360 Images
Mentah Folder has 360 Images
Images Folder has 1080 Images
```

In [4]:
```
# Parameter
base_dir = r'train_data' #direktori folder dataset
img_size = 224
batch = 10
validation_split = 0.1
```

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

In [5]:
```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
)
```

```
Found 1080 files belonging to 3 classes.
```

In [6]:
```
class_names = dataset.class_names
print("Class Names:", class_names)
```

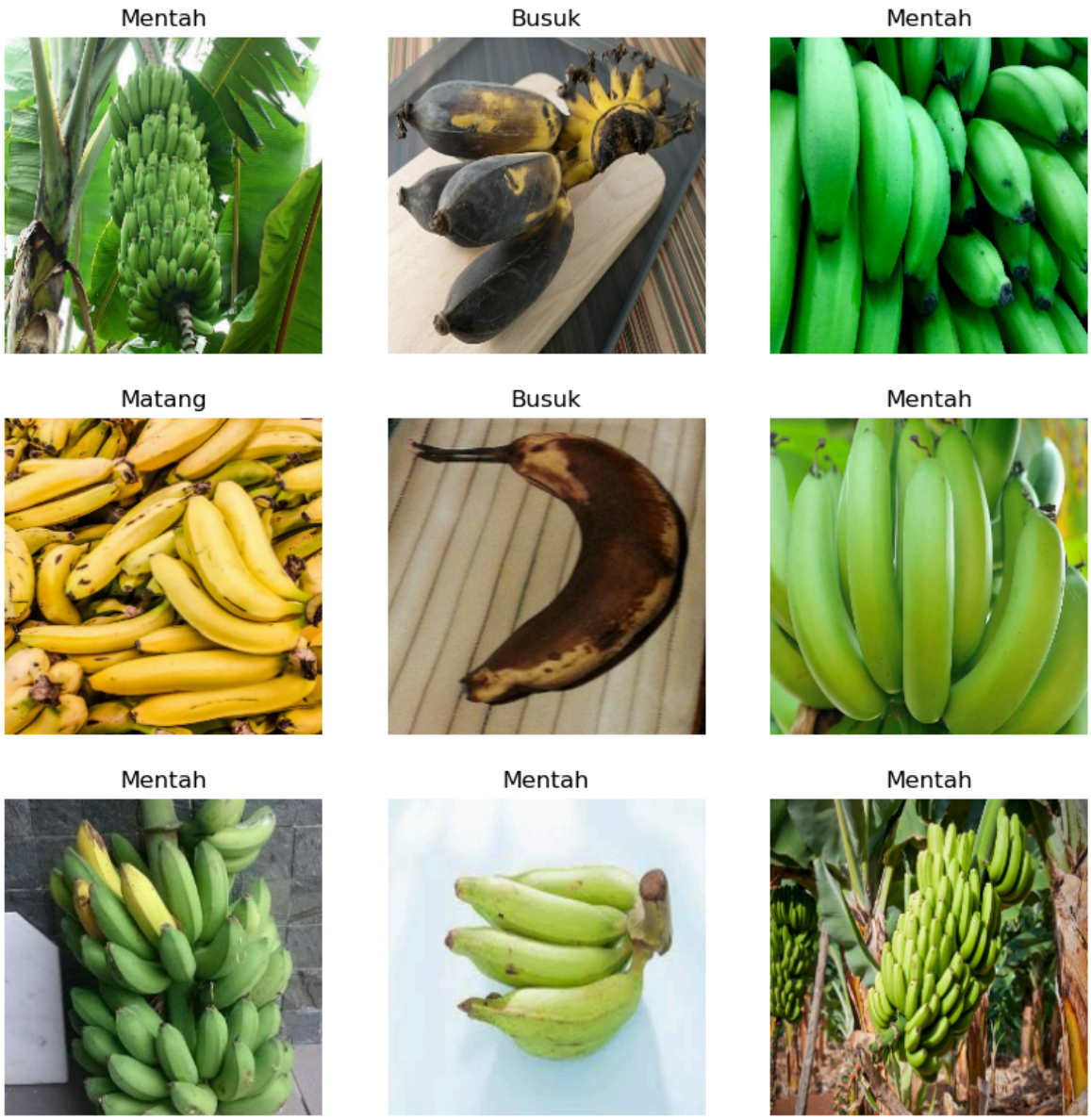
```
Class Names: ['Busuk', 'Matang', 'Mentah']
```

In [7]:
```
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: 108
Train Images: 98
Validation Images: 10
```

In [8]:
```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count) #mengambil data validasi

validation_split = 0.1
train_dataset = dataset.take(train_count)
val_dataset = dataset.skip(train_count)
val_dataset = dataset.take(val_count)
```

In [9]:
```
import matplotlib.pyplot as plt

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


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



In [10]:
```
import numpy as np

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

```
(10, 224, 224, 3)
```

In [38]:
```
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))
#Lihat data setelah di augmentasi
```

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

```
C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\prepr
ocessing\tf_data_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Input(shap
e)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

In [39]:

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, SeparableConv2D, MaxPooling2

model = Sequential([
    data_augmentation,
    Rescaling(1./255, input_shape=(img_size, img_size, 3)),

    # Block 1 (Depthwise Separable Convolutions)
    SeparableConv2D(32, (3, 3), activation='relu', padding='same', strides=2),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
```

```
    # Block 2
    SeparableConv2D(64, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Block 3
    SeparableConv2D(128, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Block 4
    SeparableConv2D(256, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Block 5
    SeparableConv2D(512, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Global Average Pooling (instead of Flatten)
    GlobalAveragePooling2D(),

    # Fully Connected Layers
    Dense(1024, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')  # Output layer for 3 classes
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',  # Change this to sparse_c
              metrics=['accuracy'])




model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
sequential_2 (Sequential)	(None, 224, 224, 3)	0
rescaling_1 (Rescaling)	(None, 224, 224, 3)	0
separable_conv2d_5 (SeparableConv2D)	(None, 112, 112, 32)	155
batch_normalization_5 (BatchNormalization)	(None, 112, 112, 32)	128
max_pooling2d_5 (MaxPooling2D)	(None, 56, 56, 32)	0
separable_conv2d_6 (SeparableConv2D)	(None, 56, 56, 64)	2,400
batch_normalization_6 (BatchNormalization)	(None, 56, 56, 64)	256
max_pooling2d_6 (MaxPooling2D)	(None, 28, 28, 64)	0
separable_conv2d_7 (SeparableConv2D)	(None, 28, 28, 128)	8,896
batch_normalization_7 (BatchNormalization)	(None, 28, 28, 128)	512
max_pooling2d_7 (MaxPooling2D)	(None, 14, 14, 128)	0
separable_conv2d_8 (SeparableConv2D)	(None, 14, 14, 256)	34,176
batch_normalization_8 (BatchNormalization)	(None, 14, 14, 256)	1,024
max_pooling2d_8 (MaxPooling2D)	(None, 7, 7, 256)	0
separable_conv2d_9 (SeparableConv2D)	(None, 7, 7, 512)	133,888
batch_normalization_9 (BatchNormalization)	(None, 7, 7, 512)	2,048
max_pooling2d_9 (MaxPooling2D)	(None, 3, 3, 512)	0
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 512)	0
dense_2 (Dense)	(None, 1024)	525,312
dropout_1 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 3)	3,075

Total params: 711,870 (2.72 MB)

Trainable params: 709,886 (2.71 MB)

Non-trainable params: 1,984 (7.75 KB)

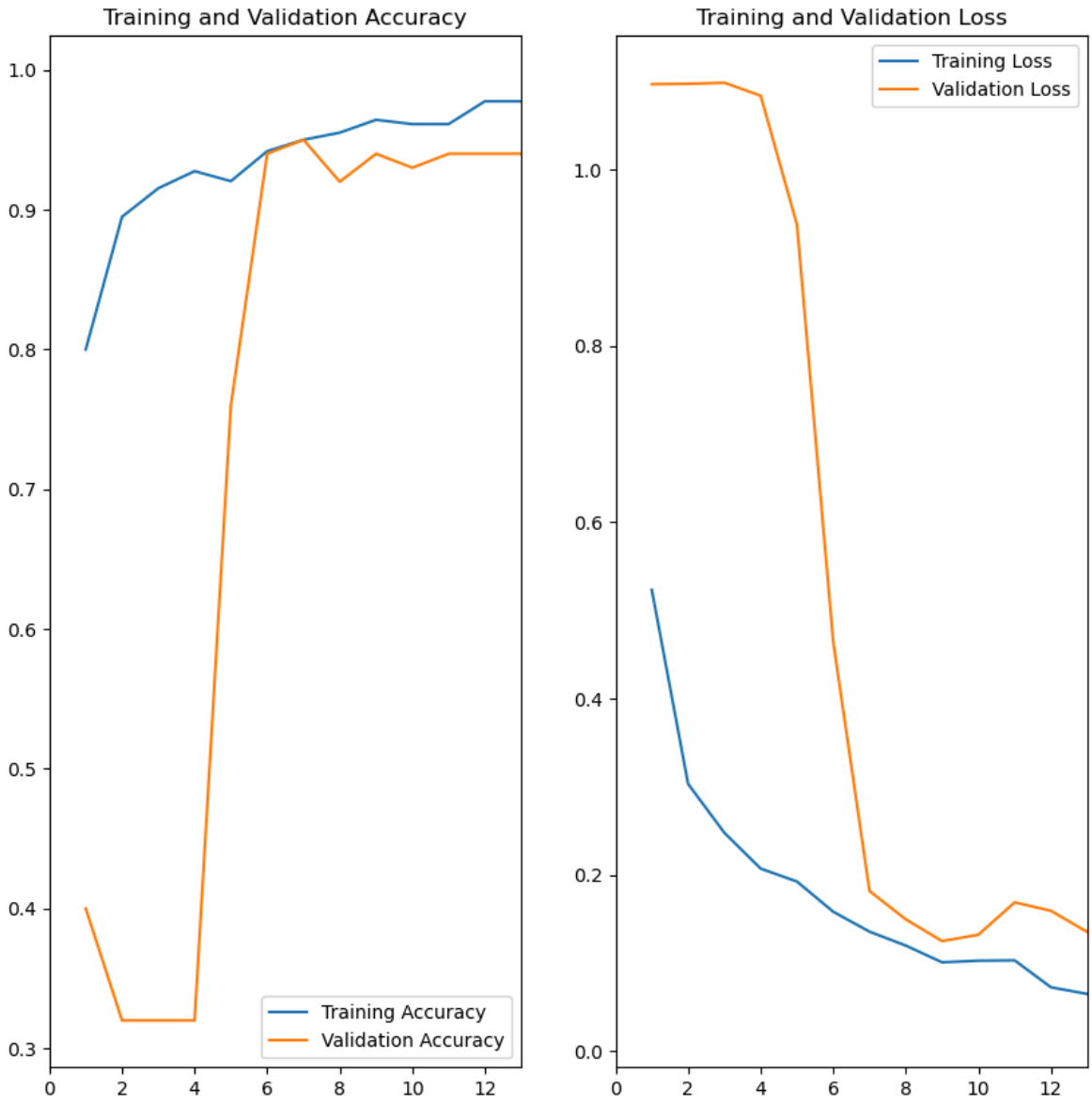
In [40]:
```
from tensorflow.keras.optimizers import RMSprop, Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from sklearn.utils.class_weight import compute_class_weight
import numpy as np


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


reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.1,
    patience=10,
    min_lr=0.00001,
    cooldown=2
)

early_stopping = EarlyStopping(
    monitor='val_accuracy',
    patience=15,
    mode='max',
    restore_best_weights=True,
    min_delta=0.001
)


y_train = []
for images, labels in train_ds:
    y_train.extend(labels.numpy())

class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(
class_weight_dict = dict(zip(np.unique(y_train), class_weights))


history = model.fit(
    train_ds,
    epochs=60,
    validation_data=val_ds,
    callbacks=[early_stopping, reduce_lr],
    class_weight=class_weight_dict
)
```

```
Epoch 1/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 12s 65ms/step - accuracy: 0.6941 - loss: 0.7350 - val_
accuracy: 0.4000 - val_loss: 1.0973 - learning_rate: 1.0000e-04
Epoch 2/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.8822 - loss: 0.3466 - val_a
ccuracy: 0.3200 - val_loss: 1.0977 - learning_rate: 1.0000e-04
Epoch 3/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9176 - loss: 0.2387 - val_a
ccuracy: 0.3200 - val_loss: 1.0989 - learning_rate: 1.0000e-04
Epoch 4/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9212 - loss: 0.2205 - val_a
ccuracy: 0.3200 - val_loss: 1.0842 - learning_rate: 1.0000e-04
Epoch 5/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9048 - loss: 0.2116 - val_a
ccuracy: 0.7600 - val_loss: 0.9379 - learning_rate: 1.0000e-04
Epoch 6/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9450 - loss: 0.1611 - val_a
ccuracy: 0.9400 - val_loss: 0.4643 - learning_rate: 1.0000e-04
Epoch 7/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 62ms/step - accuracy: 0.9455 - loss: 0.1306 - val_a
ccuracy: 0.9500 - val_loss: 0.1816 - learning_rate: 1.0000e-04
Epoch 8/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9612 - loss: 0.1035 - val_a
ccuracy: 0.9200 - val_loss: 0.1495 - learning_rate: 1.0000e-04
Epoch 9/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9708 - loss: 0.0903 - val_a
ccuracy: 0.9400 - val_loss: 0.1248 - learning_rate: 1.0000e-04
Epoch 10/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9691 - loss: 0.0886 - val_a
ccuracy: 0.9300 - val_loss: 0.1321 - learning_rate: 1.0000e-04
Epoch 11/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9564 - loss: 0.1193 - val_a
ccuracy: 0.9400 - val_loss: 0.1688 - learning_rate: 1.0000e-04
Epoch 12/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9845 - loss: 0.0667 - val_a
ccuracy: 0.9400 - val_loss: 0.1592 - learning_rate: 1.0000e-04
Epoch 13/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 62ms/step - accuracy: 0.9719 - loss: 0.0823 - val_a
ccuracy: 0.9400 - val_loss: 0.1351 - learning_rate: 1.0000e-04
Epoch 14/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 66ms/step - accuracy: 0.9776 - loss: 0.0613 - val_a
ccuracy: 0.9100 - val_loss: 0.2254 - learning_rate: 1.0000e-04
Epoch 15/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 66ms/step - accuracy: 0.9753 - loss: 0.0636 - val_a
ccuracy: 0.9300 - val_loss: 0.1784 - learning_rate: 1.0000e-04
Epoch 16/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 63ms/step - accuracy: 0.9717 - loss: 0.0579 - val_a
ccuracy: 0.9400 - val_loss: 0.1181 - learning_rate: 1.0000e-04
Epoch 17/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 63ms/step - accuracy: 0.9784 - loss: 0.0509 - val_a
ccuracy: 0.9500 - val_loss: 0.1386 - learning_rate: 1.0000e-04
Epoch 18/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 64ms/step - accuracy: 0.9904 - loss: 0.0389 - val_a
ccuracy: 0.9500 - val_loss: 0.1103 - learning_rate: 1.0000e-04
Epoch 19/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 61ms/step - accuracy: 0.9917 - loss: 0.0330 - val_a
ccuracy: 0.9500 - val_loss: 0.1539 - learning_rate: 1.0000e-04
Epoch 20/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 56ms/step - accuracy: 0.9869 - loss: 0.0374 - val_a
ccuracy: 0.9400 - val_loss: 0.1386 - learning_rate: 1.0000e-04
```

```
Epoch 21/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 58ms/step - accuracy: 0.9880 - loss: 0.0311 - val_a
ccuracy: 0.9400 - val_loss: 0.1778 - learning_rate: 1.0000e-04
Epoch 22/60
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 59ms/step - accuracy: 0.9927 - loss: 0.0282 - val_a
ccuracy: 0.9200 - val_loss: 0.1864 - learning_rate: 1.0000e-04
```

In [41]:

```
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 Accura
plt.legend(loc='lower right')
plt.xlim(0, 13)
plt.title('Training and Validation Accuracy')


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



In [42]:
```
# Save the model in .keras format
model.save('MOBILENET.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `ker
as.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 `ke
ras.saving.save_model(model, 'my_model.keras')`.
```

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


model = load_model(r'MOBILENET.h5')
class_names = ['Busuk', 'Matang', 'Mentah']

def classify_images(image_path, save_path='predicted_image.jpg'):
    try:

        input_image = tf.keras.utils.load_img(image_path, target_size=(224, 224)
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)
```

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

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

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

        return f"Prediksi: {class_names[class_idx]} dengan confidence {confidenc
    except Exception as e:
        return f"Terjadi kesalahan: {e}"
result = classify_images(r'Prediksi\Matang\matang (8).jpg', save_path='matang.jp
print(result)
```

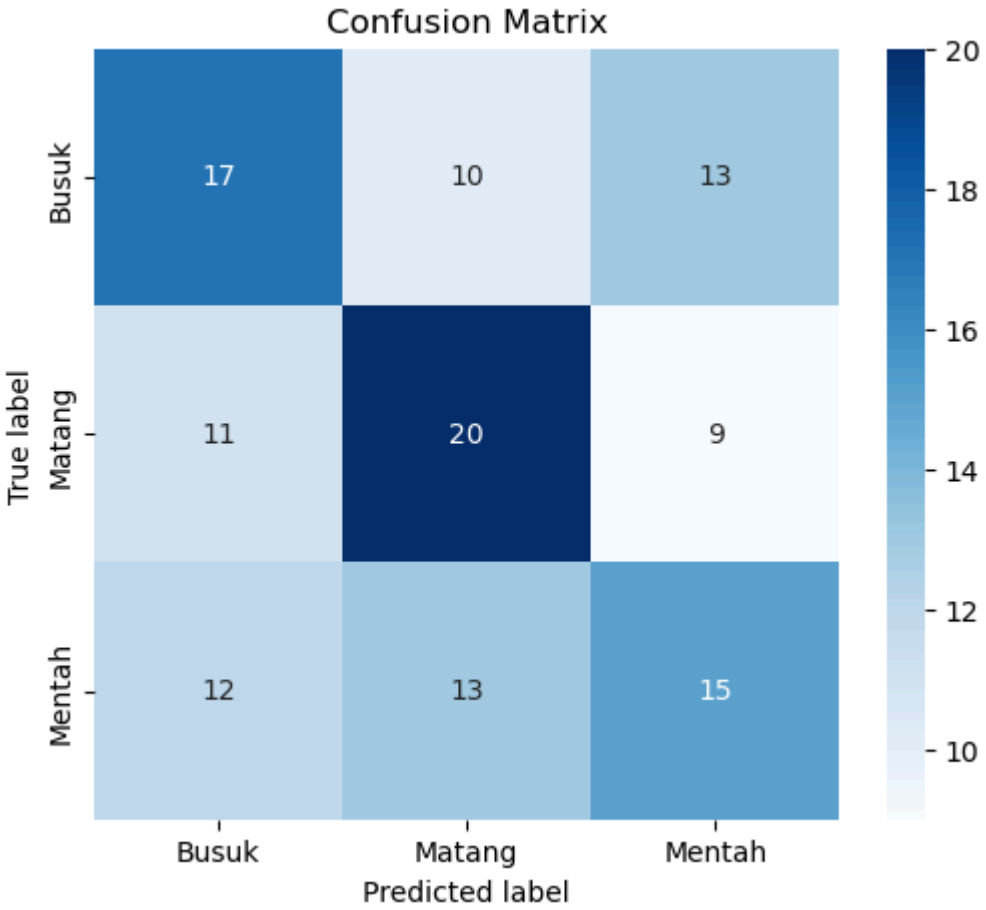
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be b
uilt. `model.compile_metrics` will be empty until you train or evaluate the mode
l.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 223ms/step
Prediksi: Matang
Confidence: 57.15%
Prediksi: Matang dengan confidence 57.15%. Gambar asli disimpan di matang.jpg.
```

In [53]:

```
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'test_data',
    labels='inferred',
    label_mode='categorical',  # Menghasilkan label dalam bentuk one-hot encodin
    batch_size=32,
    image_size=(224, 224)
)


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

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

```
Found 120 files belonging to 3 classes.
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
```



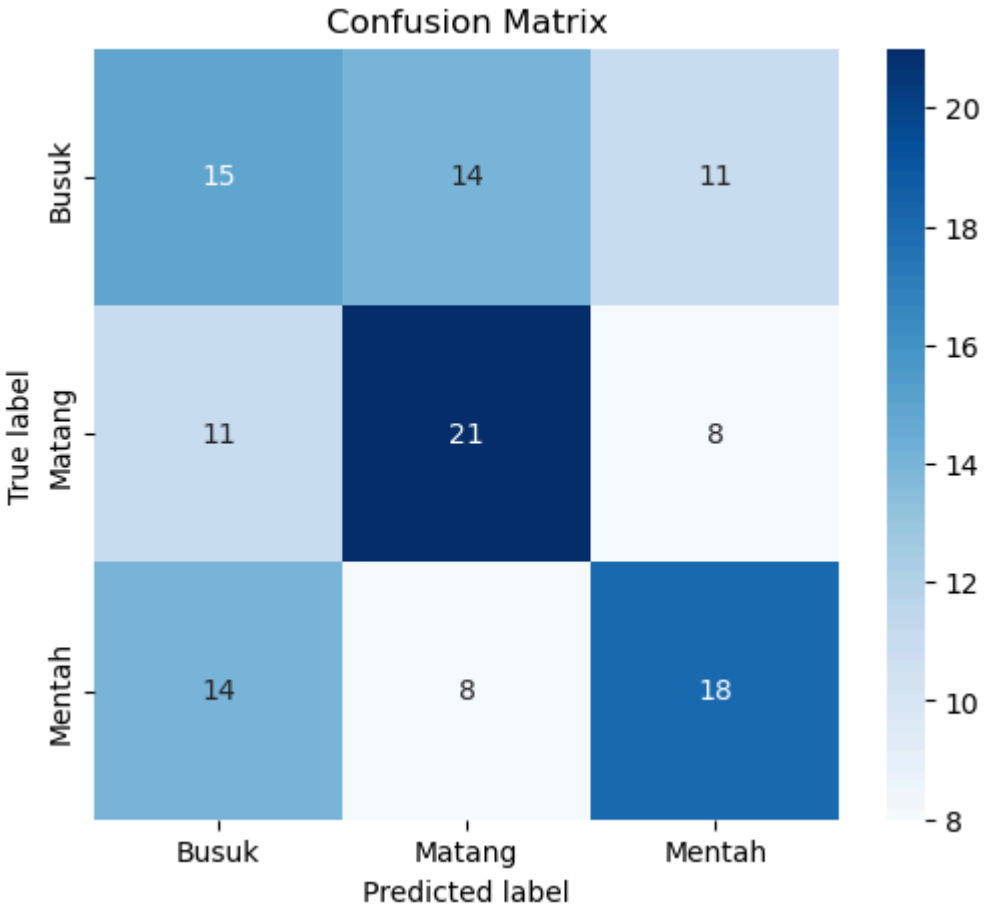

```
Confusion Matrix:
 [[17 10 13]
 [11 20  9]
 [12 13 15]]
Accuracy: 0.43333333333333335
Precision: [0.425      0.46511628 0.40540541]
Recall: [0.425 0.5   0.375]
F1 Score: [0.425      0.48192771 0.38961039]
```

In [88]:

```
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'test_data',
    labels='inferred',
    label_mode='categorical',  # Menghasilkan label dalam bentuk one-hot encodin
    batch_size=32,
    image_size=(224, 224)
)


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

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

```
Found 120 files belonging to 3 classes.
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
```



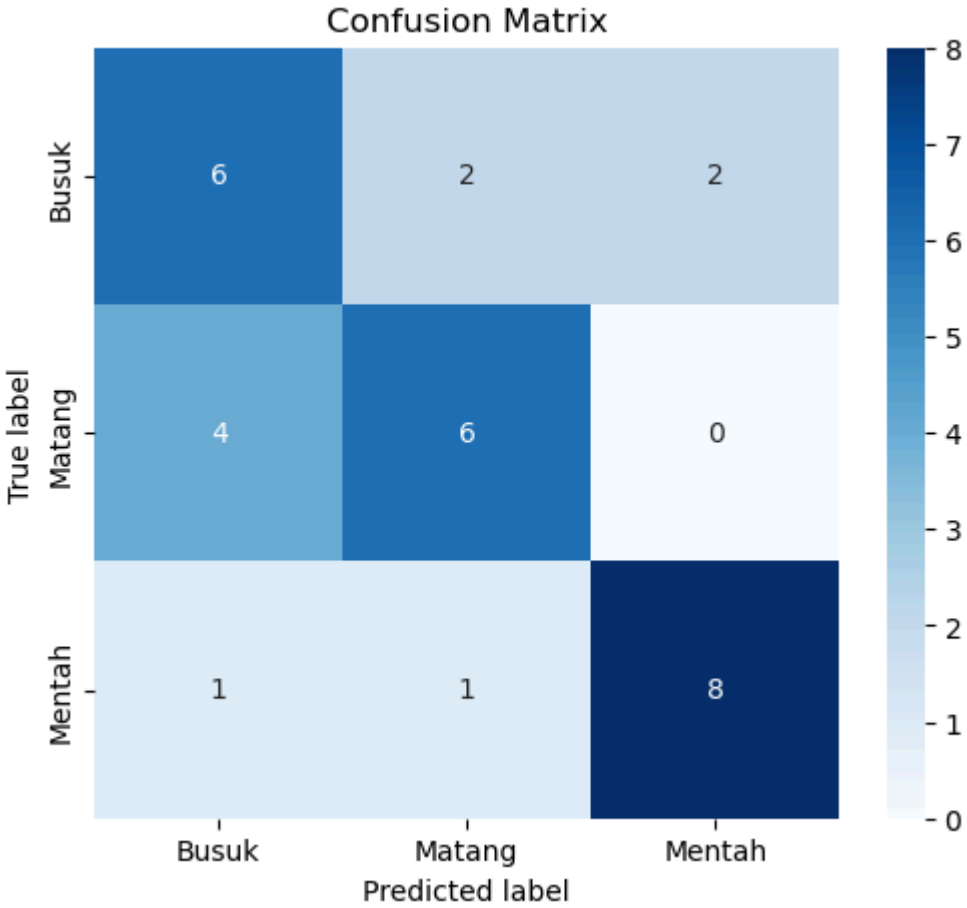

```
Confusion Matrix:
 [[15 14 11]
 [11 21  8]
 [14  8 18]]
Accuracy: 0.45
Precision: [0.375      0.48837209 0.48648649]
Recall: [0.375 0.525 0.45 ]
F1 Score: [0.375      0.5060241  0.46753247]
```

In [49]:

```python
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'Prediksi',
    labels='inferred',
    label_mode='categorical',  # Menghasilkan label dalam bentuk one-hot encodin
    batch_size=32,
    image_size=(224, 224)
)


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

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

```
Found 30 files belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
```



```
Confusion Matrix:
 [[6 2 2]
 [4 6 0]
 [1 1 8]]
Accuracy: 0.6666666666666666
Precision: [0.54545455 0.66666667 0.8       ]
Recall: [0.6 0.6 0.8]
F1 Score: [0.57142857 0.63157895 0.8       ]
```

Nama : Angelo Khara Starggang

NPM : 2207081

Nama : Dani Gerald Rajagukguk

NPM : 2207052

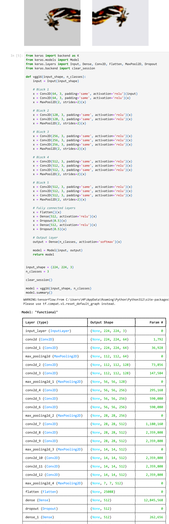
Kelompok S8 : Transformer

Arsitektur : VGG 16

```
import tensorflow as tf
import numpy as np
import matplotlib as mpl
import os

data_dir = r"/kaggle/input"
img_size = 224
batch_size = 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_size,
    subset="training",

    validation_split=validation_split,
    subset="validation"

remaining_ds = dataset.skip(train_count)

class_names = train_ds.class_names

print("Total Images: {total_count}")
print("Training Images: {train_count}")
print("Validation Images: {val_count}")
print("Batch size: {batch_size}")
print("Total Kelas: {len(class_names)}")
Found 1000 files belonging to 3 classes.
Found 1000 files belonging to 3 classes.
Using 913 files for training.
Found 1000 files belonging to 3 classes.
Using 913 files for validation.
Total Images: 10
Training Images: 9
Validation Images: 1
Batch size: 32
Total Kelas: 3
import matplotlib.pyplot as plt

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

for images, labels in train_ds.take(1):
    for i in range(9):
        ax = 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_np = images.numpy()
    print(images_np.shape)

(32, 224, 224, 3)

from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt

Tensor = 32 data BATCHNO
img_shape = (224, 224, 3) # (height, width, channels)
num_ch = img_shape[2] # channels

data_augmentation = Sequential([
    layers.InputLayer(shape=(224, 224, 3)),
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.2),
    layers.RandomZoom(height_factor=0.2, width_factor=0.2),
    layers.RandomHeight(factor=0.2),
    layers.RandomContrast(factor=0.2),
    layers.RandomTranslation(height_factor=0.2, width_factor=0.2),
    layers.GaussianNoise(0.05), # ringan saja
    layers.RandomCrop(height=224, width=224, name="final_crop"
])

for images, labels in train_ds.take(1):
    images_np = images.numpy()
    val_count = len(images_np) # ambil jumlah batch
    print("Total Images: {train_count}")
    print("Validation Images: {val_count}")

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

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

C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\preprocessing\tf_data_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(*args,
Total Images: 10
Validation Images: 100

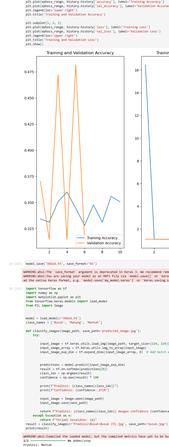


```
C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\preprocessing\tf_data_layer.py:19: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(*args,
Total Images: 10
Validation Images: 100

import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPool2D, Dropout
from tensorflow.keras.mobile import clear_session

def vgg16(input_shape, n_classes):
    inpt = Input(input_shape)

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

    # Block 2
    x = Conv2D(128, 3, padding='same', activation='relu')(x)
    x = Conv2D(128, 3, padding='same', activation='relu')(x)
    x = MaxPool2D(2, strides=2)(x)

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

    # Block 4
    x = Conv2D(512, 3, padding='same', activation='relu')(x)
    x = Conv2D(512, 3, padding='same', activation='relu')(x)
    x = Conv2D(512, 3, padding='same', activation='relu')(x)
    x = MaxPool2D(2, strides=2)(x)

    # Block 5
    x = Conv2D(512, 3, padding='same', activation='relu')(x)
    x = Conv2D(512, 3, padding='same', activation='relu')(x)
    x = Conv2D(512, 3, padding='same', activation='relu')(x)
    x = MaxPool2D(2, strides=2)(x)

    # Fully connected layers
    x = Flatten()(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)

    # Output layer
    output = Dense(n_classes, activation='softmax')(x)

    model = Model(inpt, output)
    return model

clear_session()

model = vgg16(input_shape, n_classes)
model.summary()

WARNING:tensorflow:From C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\keras\src\backend\common\global_state.py:82: The name tf.reset_default_graph is deprecated. Please use tf.compat.v1.reset_default_graph instead.

Model: "functional"
```

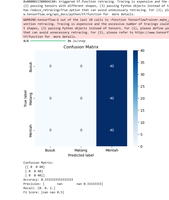
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36,928
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73,856
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147,584
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295,168
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590,080
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590,080
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_7 (Conv2D)	(None, 28, 28, 512)	1,180,160
conv2d_8 (Conv2D)	(None, 28, 28, 512)	2,359,808
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2,359,808
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_10 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 512)	12,845,568
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262,656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 3)	1,539

Total params: 27,824,451 (106.16 MB)
Trainable params: 27,824,451 (106.16 MB)
Non-trainable params: 0 (0.00 B)

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

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

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

history= model.fit(train_ds,
                   epochs=10,
                   validation_data=val_ds,
                   callbacks=[early_stopping]
Epoch 1/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 285s 311ms/step - accuracy: 0.7586 - loss: 46.1622 - val_accuracy: 0.5700 - val_loss: 2.8975
Epoch 2/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 3/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 4/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 5/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 6/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 7/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 8/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 9/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992
Epoch 10/10
913/913 ━━━━━━━━━━━━━━━━━━━━ 284s 311ms/step - accuracy: 0.7237 - loss: 1.1061 - val_accuracy: 0.5100 - val_loss: 1.0992

history_range = range(1, len(history.history['accuracy']) + 1)
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(history_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(history_range, history.history['loss'], label='Training Loss')
plt.plot(history_range, history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```




```
model.save('VGG16.h5', save_format='h5')

WARNING:absl:You are saving your model as an HDF5 file via `model.save()`. This is deprecated in Keras 3. We recommend removing this argument as it can be inferred from the file path. Recommend `save_format=keras` or `save_format=tf` or `save_format=h5` or `save_format=keras/v3`. Keras 3 will default to saving in the Keras v3 format.
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 as mpl
import os, sys, shutil, glob, random, datetime, time, math, gc, warnings, pathlib, psutil, cv2, PIL, tqdm, joblib, sklearn, scipy
from PIL import Image

model = load_model('VGG16.h5')
class_names = ['Busuk', 'Matang', 'Mentah']

def classify_image(image_path, base_path=r'prediction_image_2'):
    try:
        full_path = f'{base_path.rstrip(\'\\\')}\{image_path}'
        img = tf.keras.utils.load_img(full_path, target_size=(224, 224))
        img_array = tf.keras.utils.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0) # Add batch dimension

        predictions = model.predict(img_array, verbose=0)
        result = tf.nn.softmax(predictions[0])
        class_idx = np.argmax(result)
        confidence = float(result[class_idx])

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

        input_image = image.load_img(full_path)
        input_image.save(save_path)

        return {"Predict": class_names[class_idx], "Angka": confidence, "Path": save_path}
    except Exception as e:
        print(f"Error processing {image_path}: {e}")
        return None
model = classify_image("Mentah/Mentah (2).jpg")
print(model)

WARNING:tensorflow: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.
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000021080BC2BC0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option to avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.
Prediction: Mentah
Confidence: 0.99
{'Predict': Mentah, 'Angka': 0.9935, 'Path': 'predict_image_2\Mentah (2).jpg'}

import tensorflow as tf
import numpy as np
import matplotlib as mpl
import os

test_dir = tf.keras.preprocessing.image_dataset_from_directory(
    r"/kaggle/input",
    labels='inferred',
    label_mode='categorical',
    image_size=(224, 224),
    batch_size=32,
    shuffle=False
)

y_pred = model.predict(test_dir)
y_pred_classes = np.argmax(y_pred, axis=1)

true_labels = []
for _, labels in test_dir:
    true_labels.extend(np.argmax(labels, axis=1).numpy())
true_labels = np.concatenate(true_labels)

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

fn_rate = 1 - tf.cast(recall * tf.reduce_sum(conf_mat, axis=1)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=['Busuk', 'Matang', 'Mentah'], yticklabels=['Busuk', 'Matang', 'Mentah'])
plt.title('Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()

# Menampilkan hasil
print("Confusion Matrix:")
print(conf_mat.numpy()
print("Accuracy: {accuracy:.4f}")
print("Precision: {precision.numpy()}")
print("Recall: {recall.numpy()}")
print("False Negative Rate: {fn_rate.numpy()}")

Found 120 files belonging to 3 classes.
WARNING:tensorflow:5 out of the last 13 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000021080BC2BC0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option to avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.
WARNING:tensorflow:5 out of the last 18 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000021080BC2BC0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option to avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.
WARNING:tensorflow:5 out of the last 23 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000021080BC2BC0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option to avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.
Confusion Matrix:
[[ 0 40 0]
[ 0 40 0]
[ 0 40 0]]
Accuracy: 0.3333
Precision: [0. 0. 0.]
Recall: [0. 0. 1.]
False Negative Rate: [1. 1. 0.]
```



```python
import streamlit as st
from streamlit_extras.let_it_rain import rain
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image
from datetime import datetime, timedelta
import time
import base64
from pathlib import Path
from st_social_media_links import SocialMediaIcons

# Load the pre-trained model
model = load_model(r'E:\streamlit\BestModel_MOBILENET_Transformer.h5')  # Update with your model path
class_names = ['Busuk', 'Matang', 'Mentah']

# Function to preprocess and classify image
def classify_image(image_path):
    try:
        # Load and preprocess the image
        input_image = tf.keras.utils.load_img(image_path, target_size=(180, 180))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)

        # Predict using the model
        predictions = model.predict(input_image_exp_dim)
        result = tf.nn.softmax(predictions[0])  # Apply softmax for probability

        # Get class with highest confidence
        class_idx = np.argmax(result)
        confidence_scores = result.numpy()
        return class_names[class_idx], confidence_scores
    except Exception as e:
        return "Error", str(e)

# Function to create a custom progress bar
def custom_progress_bar(confidence, color1, color2, color3):
    percentage1 = confidence[0] * 100  # Confidence for class 0 (Busuk)
    percentage2 = confidence[1] * 100  # Confidence for class 1 (Matang)
    percentage3 = confidence[2] * 100  # Confidence for class 2 (Mentah)
    progress_html = f"""
    <div style="border: 1px solid #ddd; border-radius: 5px; overflow: hidden; width: 100%; font-size: 14px;">
        <div style="width: {percentage1:.2f}%; background: {color1}; color: white; text-align: center; height: 24px; float: left;">
            {percentage1:.2f}% Busuk
        </div>
        <div style="width: {percentage2:.2f}%; background: {color2}; color: black; text-align: center; height: 24px; float: left;">
            {percentage2:.2f}% Matang
        </div>
        <div style="width: {percentage3:.2f}%; background: {color3}; color: black; text-align: center; height: 24px; float: left;">
            {percentage3:.2f}% Mentah
        </div>
    </div>
    """
    st.sidebar.markdown(progress_html, unsafe_allow_html=True)

def get_base64(bin_file):
    with open(bin_file, 'rb') as f:
        data = f.read()
    return base64.b64encode(data).decode()

def set_background(jpg_file):
    bin_str = get_base64(jpg_file)
    page_bg_img = '''
    <style>
    .stApp {
    background-image: url("%s");
    background-size: cover;
    }
    </style>
    ''' % bin_str
    st.markdown(page_bg_img, unsafe_allow_html=True)

# Function to apply Fire effect
def run_fire_animation():
    rain(emoji="🔥", font_size=40, falling_speed=10, animation_length="infinite")

# Run fire animation
run_fire_animation()

st.balloons()

# Change Background Streamlit
set_background(r"E:\streamlit\background1.gif")

# Add custom background with snow animation
background = """
<style>
/* Set full-page background */
# [data-testid="stAppViewContainer"] {
#     background: linear-gradient(to bottom, #FF0000, #00FF00); /* Red to Green */
#     background-size: cover;
#     background-attachment: fixed;
#     color: white; /* Text color */
#     font-family: Arial, sans-serif;
# }
```

```
/* Hide default Streamlit styling for a cleaner look */
[data-testid="stSidebar"] {
    background: #fbb99f;
    color: white;
}

/* Perbaiki warna teks judul agar tetap terlihat */
h1, h2, h3, h4, h5, h6, p {
    color: white;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.8); /* Efek bayangan */
}

/* Create fire animation */
.fire {
    color: #red;
    font-size: 1.5em;
    position: absolute;
    top: -10%;
    animation: fire 10s linear infinite;
    opacity: 0.8;
    z-index: 10;
}

@keyframes fire {
    0% { transform: translateY(0); }
    100% { transform: translateY(100vh); }
}

.fire:nth-child(1) { left: 10%; animation-delay: 0s; }
.fire:nth-child(2) { left: 20%; animation-delay: 2s; }
.fire:nth-child(3) { left: 30%; animation-delay: 4s; }
.fire:nth-child(4) { left: 40%; animation-delay: 6s; }
.fire:nth-child(5) { left: 50%; animation-delay: 8s; }
.fire:nth-child(6) { left: 60%; animation-delay: 1s; }
.fire:nth-child(7) { left: 70%; animation-delay: 3s; }
.fire:nth-child(8) { left: 80%; animation-delay: 5s; }
.fire:nth-child(9) { left: 90%; animation-delay: 7s; }
.fire:nth-child(10) { left: 100%; animation-delay: 9s; }
</style>
"""

# Display the background animation and fire
st.markdown(background, unsafe_allow_html=True)

title_html = """
<div style="text-align: center; color: #fbb99f; text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.8); font-size: 50px; font-weight: bold;">
    🔥 Prediksi Kematangan Buah Pisang - Transformer 🔥
</div>
"""
st.markdown(title_html, unsafe_allow_html=True)

file_uploader_style = """
<style>
/* Tambahkan padding untuk memperbesar area komponen */
div[data-testid="stFileUploader"] {
    margin-top: 20px;
    margin-bottom: 20px;
    padding: 20px;
    border: 2px dashed #FF4500; /* Tambahkan border dengan warna */
    border-radius: 10px; /* Border radius untuk sudut melengkung */
    background-color: #fbb99f; /* Warna latar belakang (peach puff) */
}
</style>
"""

# Terapkan styling ke file uploader
st.markdown(file_uploader_style, unsafe_allow_html=True)

# Upload multiple files in the main page
uploaded_files = st.file_uploader(
    "Unggah Gambar (Beberapa diperbolehkan)",
    type=["jpg", "png", "jpeg"],
    accept_multiple_files=True
)

if uploaded_files:
    for uploaded_file in uploaded_files:
        image = Image.open(uploaded_file)
        st.sidebar.image(image, caption=f"{uploaded_file.name}", use_column_width=True)

# Style for the prediction button
style_button = """
<style>
.button-prediksi {
    display: block;
    margin: 0 auto;
    text-align: center;
}
</style>
"""
st.markdown(style_button, unsafe_allow_html=True)

# Sidebar for prediction button and results
col1, col2, col3 = st.sidebar.columns([1, 1, 1])
with col2:
    if st.button("Prediksi"):
```

```python
        st.snow()
        if uploaded_files:
            st.sidebar.write("### 🎁 Hasil Prediksi")
            with st.spinner('Memprediksi...'):
                for uploaded_file in uploaded_files:
                    with open(uploaded_file.name, "wb") as f:
                        f.write(uploaded_file.getbuffer())

                    # Perform prediction
                    label, confidence = classify_image(uploaded_file.name)

                    if label != "Error":
                        # Define colors for the bar and label
                        primary_color = "#000000"  # Black for "Busuk"
                        secondary_color = "#FFFF00"  # Yellow for "Matang"
                        tertiary_color = "00FF00"  # Green for "Mentah"

                        if label == "Busuk":
                            label_color = primary_color
                        elif label == "Matang":
                            label_color = secondary_color
                        elif label == "Mentah":
                            label_color = tertiary_color

                        # Display prediction results
                        st.sidebar.write(f"*Nama File:* {uploaded_file.name}")
                        st.sidebar.markdown(f"<h4 style='color: {label_color};'>Prediksi: {label}</h4>", unsafe_allow_html=True)

                        # Display confidence scores
                        st.sidebar.write("*Confidence:*")
                        for i, class_name in enumerate(class_names):
                            st.sidebar.write(f"- {class_name}: {confidence[i] * 100:.2f}%")

                        # Display custom progress bar
                        custom_progress_bar(confidence, primary_color, secondary_color, tertiary_color)

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

# Preview images in the main page
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)

# Tambahkan copyright di bagian bawah
copyright_html = """
<div style="text-align: center; margin-top: 5px; font-size: 20px; font-wight: bold; color: #fbb99f; opacity: 1;">
    © 2024 Transformer. All Rights Reserved. <br>
    Frans Daniel Rajagukguk / 220711826 <br>
    Angello Khara Sitanggang / 220711833 <br>
    Davin Gilbert Natanael / 220711841 <br>
    Emanuel Enrico Anindya Wibawa / 220711890 <br>
    Archipera Rari Dyaksa / 220711891 <br>
</div>
"""
st.markdown(copyright_html, unsafe_allow_html=True)

main_social_media_html = """
<div style="text-align: center; margin-top: 10px;">
    <a href="https://github.com/ANTIMOLE/Project-UAS-Sementara" target="_blank">
        <img src="https://upload.wikimedia.org/wikipedia/commons/9/91/Octicons-mark-github.svg" alt="GitHub" style="width: 40px; margin: 10px;">
    </a>
    <a href="https://youtu.be/dQw4w9WgXcQ?si=xq0SADs2r9-txQOj" target="_blank">
        <img src="https://upload.wikimedia.org/wikipedia/commons/a/a5/Instagram_icon.png" alt="Instagram" style="width: 40px; margin: 10px;">
    </a>
</div>
"""
st.markdown(main_social_media_html, unsafe_allow_html=True)
```