

```

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

def add_bg_from_url():
    st.markdown(
        f"""
        <style>
        .stApp {{
            background-image: linear-gradient(rgba(0,0,0,0.2), rgba(0,0,0,2)), url("https://i.pinimg.com/736x/c9/6d/60/c96d601d5ba32d581f40a8a06d1e0971.jpg");
            background-attachment: fixed;
            background-size: cover;
        }}

        /* Tambahan style untuk membuat konten lebih readable */
        .main {{
            background-color: rgba(255, 255, 255, 0.85);
            padding: 20px;
            border-radius: 10px;
        }}

        .sidebar .sidebar-content {{
            background-color: rgba(255, 255, 255, 0.9);
        }}
        </style>
        """,
        unsafe_allow_html=True
    )

add_bg_from_url()

model = load_model(r"D:\Kuliah\Semester 5\ML\UAS\BestModel_MobilNetCNN_Seaborn.h5")

class_names = ['PaprikaHijau', 'PaprikaKuning', 'PaprikaMerah']

def preprocess_image(image_array):
    return tf.cast(image_array, tf.float32) / 255.0

def classify_image(image_path):
    try:
        input_image = tf.keras.utils.load_img(image_path, target_size=(180, 180))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)

        predictions = model.predict(input_image_exp_dim)
        result = tf.nn.softmax(predictions[0])

        class_idx = np.argmax(result)
        confidence_scores = result.numpy()

        return class_names[class_idx], confidence_scores
    except Exception as e:
        return "Error", str(e)

def custom_progress_bar(confidence, color_map):
    percentage1 = confidence[0] * 100 # Hijau
    percentage2 = confidence[1] * 100 # Kuning
    percentage3 = confidence[2] * 100 # Merah

    progress_html = f"""
    <div style="width: 100%; border: 1px solid #ddd; border-radius: 5px; overflow: hidden; width: 100%; font-size: 14px;">
    <div style="width: {percentage1:.2f}%; background: {color_map['PaprikaHijau']}; color: white; text-align: center; height: 24px; float: left;">
    {percentage1:.2f}%
    </div>
    <div style="width: {percentage2:.2f}%; background: {color_map['PaprikaKuning']}; color: white; text-align: center; height: 24px; float: left;">
    {percentage2:.2f}%
    </div>
    <div style="width: {percentage3:.2f}%; background: {color_map['PaprikaMerah']}; color: white; text-align: center; height: 24px; float: left;">
    {percentage3:.2f}%
    </div>
    </div>
    """
    st.sidebar.markdown(progress_html, unsafe_allow_html=True)

st.title("Prediksi Jenis Paprika - Seaborn")

uploaded_files = st.file_uploader("Unggah Gambar (Beberapa diperbolehkan)", 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":
                color_map = {
                    "PaprikaHijau": "#008000",
                    "PaprikaKuning": "#FFD700",
                    "PaprikaMerah": "#FF0000"
                }

                label_color = color_map.get(label, "#000000")

                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)

                st.sidebar.write(f"*Confidence: {confidence}")

```

```
display_names = ["PaprikaHijau", "PaprikaKuning", "PaprikaMerah"] # Sesuai urutan model tanpa 'Paprika'
for i, name in enumerate(display_names):
    st.sidebar.write(f"- {name}: {confidence[i] * 100:.2f}%")

custom_progress_bar(confidence, color_map)

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.")

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)
```

```

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

# Load data
data_dir = r"C:\Users\Lenovo\Documents\Kuliah - UAJY\Semester V -
2024-2025\Pembelajaran mesin dan Pembelajaran Mendalam - A\Projek UAS\
data_paprika"

# Load dataset with randomization and resizing
data = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(180, 180),
    batch_size=16
)

# Print class names
class_names = data.class_names
print("Classes:", class_names)

Found 330 files belonging to 3 classes.
Classes: ['hijau', 'kuning', 'merah']

img_size = 180
batch = 32
validation_split = 0.1

# Load dataset with specified parameters
dataset = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch
)

Found 330 files belonging to 3 classes.

total_count = len(dataset)
train_ratio = 0.8
val_ratio = 0.1
test_ratio = 0.1

train_count = int(total_count * train_ratio)
val_count = int(total_count * val_ratio)
test_count = total_count - train_count - val_count

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

```

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

Total Images: 11
Train Images: 8
Validation Images: 1
Test Images: 2

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

# Display training images
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')
plt.show()
```

kuning



shutterstock.com - 562069531

merah



merah



merah



kuning



hijau



hijau



hijau



merah



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

Image Array Shape: (32, 180, 180, 3)

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

```
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)
```

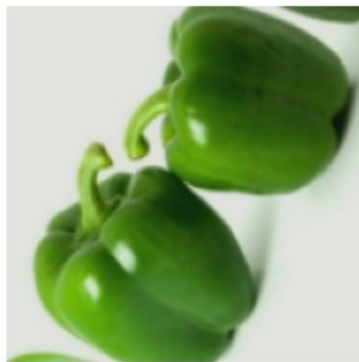
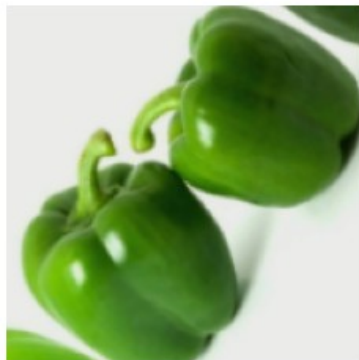
```

# Data Augmentation
data_augmentation = Sequential([
    layers.RandomFlip("horizontal_and_vertical",
input_shape=(img_size, img_size, 3)),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.15),
    layers.RandomBrightness(0.1),
    layers.RandomContrast(0.25),
    layers.GaussianNoise(0.1)
])

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

c:\Users\Lenovo\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 tensorflow.keras.backend as K
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Conv2D
from tensorflow.keras.layers import Flatten, MaxPool2D, AvgPool2D
from tensorflow.keras.layers import Concatenate, Dropout

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

        t2 = Conv2D(f[1], 1, activation='relu')(x)
```

```

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

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

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

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

    input = Input(input_shape)

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

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

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

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

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

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

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

    model = Model(input, output)
    return model

# Input shape and class configuration
input_shape = (180, 180, 3)
n_classes = 3

# Clear Keras session
K.clear_session()

```



```
# Create model
```

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

```
model.summary()
```

```
WARNING:tensorflow:From c:\Users\Lenovo\anaconda3\Lib\site-packages\keras\src\backend\common\global_state.py:73: The name tf.reset_default_graph is deprecated. Please use tf.compat.v1.reset_default_graph instead.
```

```
Model: "functional_1"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 180, 180, 3)	0	-
conv2d (Conv2D) [0]	(None, 90, 90, 64)	9,472	input_layer[0]
max_pooling2d (MaxPooling2D)	(None, 45, 45, 64)	0	conv2d[0][0]
conv2d_1 (Conv2D) [0]	(None, 45, 45, 64)	4,160	max_pooling2d[0]
conv2d_2 (Conv2D)	(None, 45, 45, 192)	110,784	conv2d_1[0][0]
max_pooling2d_1	(None, 22, 22, 192)	0	conv2d_2[0][0]

(MaxPooling2D)	192)		
conv2d_4 (Conv2D)	(None, 22, 22,	18,528	
max_pooling2d_1[0][...]	96)		
conv2d_6 (Conv2D)	(None, 22, 22,	3,088	
max_pooling2d_1[0][...]	16)		
max_pooling2d_2	(None, 22, 22,	0	
max_pooling2d_1[0][...]	(MaxPooling2D)	192)	
conv2d_3 (Conv2D)	(None, 22, 22,	12,352	
max_pooling2d_1[0][...]	64)		
conv2d_5 (Conv2D)	(None, 22, 22,	110,720	conv2d_4[0][0]
	128)		
conv2d_7 (Conv2D)	(None, 22, 22,	12,832	conv2d_6[0][0]
	32)		
conv2d_8 (Conv2D)	(None, 22, 22,	6,176	
max_pooling2d_2[0][...]	32)		
concatenate	(None, 22, 22,	0	conv2d_3[0][0],

(Concatenate)	256)		conv2d_5[0][0],
			conv2d_7[0][0],
			conv2d_8[0][0]
conv2d_10 (Conv2D)	(None, 22, 22,	32,896	concatenate[0]
[0]	128)		
conv2d_12 (Conv2D)	(None, 22, 22,	8,224	concatenate[0]
[0]	32)		
max_pooling2d_3	(None, 22, 22,	0	concatenate[0]
[0]	(MaxPooling2D)	256)	
conv2d_9 (Conv2D)	(None, 22, 22,	32,896	concatenate[0]
[0]	128)		
conv2d_11 (Conv2D)	(None, 22, 22,	221,376	conv2d_10[0][0]
	192)		
conv2d_13 (Conv2D)	(None, 22, 22,	76,896	conv2d_12[0][0]
	96)		
conv2d_14 (Conv2D)	(None, 22, 22,	16,448	
max_pooling2d_3[0][...	64)		

concatenate_1	(None, 22, 22,	0	conv2d_9[0][0],
(Concatenate)	480)		conv2d_11[0][0],
			conv2d_13[0][0],
			conv2d_14[0][0]
max_pooling2d_4	(None, 11, 11,	0	concatenate_1[0]
(MaxPooling2D)	480)		
conv2d_16 (Conv2D)	(None, 11, 11,	46,176	
max_pooling2d_4[0][...	96)		
conv2d_18 (Conv2D)	(None, 11, 11,	7,696	
max_pooling2d_4[0][...	16)		
max_pooling2d_5	(None, 11, 11,	0	
max_pooling2d_4[0][...	480)		
(MaxPooling2D)			
conv2d_15 (Conv2D)	(None, 11, 11,	92,352	
max_pooling2d_4[0][...	192)		
conv2d_17 (Conv2D)	(None, 11, 11,	179,920	conv2d_16[0][0]
	208)		
conv2d_19 (Conv2D)	(None, 11, 11,	19,248	conv2d_18[0][0]

	48)		
conv2d_20 (Conv2D)	(None, 11, 11,	30,784	
max_pooling2d_5[0][...]	64)		
concatenate_2	(None, 11, 11,	0	conv2d_15[0][0],
(Concatenate)	512)		conv2d_17[0][0],
			conv2d_19[0][0],
			conv2d_20[0][0]
conv2d_22 (Conv2D)	(None, 11, 11,	57,456	concatenate_2[0]
[0]	112)		
conv2d_24 (Conv2D)	(None, 11, 11,	12,312	concatenate_2[0]
[0]	24)		
max_pooling2d_6	(None, 11, 11,	0	concatenate_2[0]
[0]	(MaxPooling2D)	512)	
conv2d_21 (Conv2D)	(None, 11, 11,	82,080	concatenate_2[0]
[0]	160)		
conv2d_23 (Conv2D)	(None, 11, 11,	226,016	conv2d_22[0][0]
	224)		

conv2d_25 (Conv2D)	(None, 11, 11, 64)	38,464	conv2d_24[0][0]
conv2d_26 (Conv2D)	(None, 11, 11, 64)	32,832	max_pooling2d_6[0][0]
concatenate_3 (Concatenate)	(None, 11, 11, 512)	0	conv2d_21[0][0], conv2d_23[0][0], conv2d_25[0][0], conv2d_26[0][0]
conv2d_28 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]
conv2d_30 (Conv2D)	(None, 11, 11, 24)	12,312	concatenate_3[0]
max_pooling2d_7 (MaxPooling2D)	(None, 11, 11, 512)	0	concatenate_3[0]
conv2d_27 (Conv2D)	(None, 11, 11, 128)	65,664	concatenate_3[0]
conv2d_29 (Conv2D)	(None, 11, 11, 295,168)		conv2d_28[0][0]

	256)		
conv2d_31 (Conv2D)	(None, 11, 11, 64)	38,464	conv2d_30[0][0]
conv2d_32 (Conv2D)	(None, 11, 11, 64)	32,832	
max_pooling2d_7[0][...]			
concatenate_4	(None, 11, 11, 512)	0	conv2d_27[0][0], conv2d_29[0][0], conv2d_31[0][0], conv2d_32[0][0]
conv2d_34 (Conv2D)	(None, 11, 11, 144)	73,872	concatenate_4[0]
conv2d_36 (Conv2D)	(None, 11, 11, 32)	16,416	concatenate_4[0]
max_pooling2d_8	(None, 11, 11, 512)	0	concatenate_4[0]
(MaxPooling2D)			
conv2d_33 (Conv2D)	(None, 11, 11, 112)	57,456	concatenate_4[0]

conv2d_35 (Conv2D)	(None, 11, 11, 288)	373,536	conv2d_34[0][0]
conv2d_37 (Conv2D)	(None, 11, 11, 64)	51,264	conv2d_36[0][0]
conv2d_38 (Conv2D) max_pooling2d_8[0][...	(None, 11, 11, 64)	32,832	
concatenate_5 (Concatenate)	(None, 11, 11, 528)	0	conv2d_33[0][0], conv2d_35[0][0], conv2d_37[0][0], conv2d_38[0][0]
conv2d_40 (Conv2D) [0]	(None, 11, 11, 160)	84,640	concatenate_5[0]
conv2d_42 (Conv2D) [0]	(None, 11, 11, 32)	16,928	concatenate_5[0]
max_pooling2d_9 [0] (MaxPooling2D)	(None, 11, 11, 528)	0	concatenate_5[0]
conv2d_39 (Conv2D) [0]	(None, 11, 11, 135,424)		concatenate_5[0]

	256)		
conv2d_41 (Conv2D)	(None, 11, 11, 320)	461,120	conv2d_40[0][0]
conv2d_43 (Conv2D)	(None, 11, 11, 128)	102,528	conv2d_42[0][0]
conv2d_44 (Conv2D)	(None, 11, 11, 128)	67,712	
max_pooling2d_9[0][...]			
concatenate_6	(None, 11, 11, 832)	0	conv2d_39[0][0], conv2d_41[0][0], conv2d_43[0][0], conv2d_44[0][0]
max_pooling2d_10[0]	(None, 6, 6, 832)	0	concatenate_6[0]
conv2d_46 (Conv2D)	(None, 6, 6, 160)	133,280	
conv2d_48 (Conv2D)	(None, 6, 6, 32)	26,656	
max_pooling2d_11	(None, 6, 6, 832)	0	

conv2d_45 (Conv2D)	(None, 6, 6, 256)	213,248	
max_pooling2d_10[0]...			
conv2d_47 (Conv2D)	(None, 6, 6, 320)	461,120	conv2d_46[0][0]
conv2d_49 (Conv2D)	(None, 6, 6, 128)	102,528	conv2d_48[0][0]
conv2d_50 (Conv2D)	(None, 6, 6, 128)	106,624	
max_pooling2d_11[0]...			
concatenate_7	(None, 6, 6, 832)	0	conv2d_45[0][0],
(Concatenate)			conv2d_47[0][0],
			conv2d_49[0][0],
			conv2d_50[0][0]
conv2d_52 (Conv2D)	(None, 6, 6, 192)	159,936	concatenate_7[0]
conv2d_54 (Conv2D)	(None, 6, 6, 48)	39,984	concatenate_7[0]
max_pooling2d_12	(None, 6, 6, 832)	0	concatenate_7[0]
(MaxPooling2D)			
conv2d_51 (Conv2D)	(None, 6, 6, 384)	319,872	concatenate_7[0]
conv2d_53 (Conv2D)	(None, 6, 6, 384)	663,936	conv2d_52[0][0]

conv2d_55 (Conv2D)	(None, 6, 6, 128)	153,728	conv2d_54[0][0]
conv2d_56 (Conv2D)	(None, 6, 6, 128)	106,624	
max_pooling2d_12[0]...			
concatenate_8	(None, 6, 6, 1024)	0	conv2d_51[0][0], conv2d_53[0][0], conv2d_55[0][0], conv2d_56[0][0]
average_pooling2d[0] (AveragePooling2D)	(None, 4, 4, 1024)	0	concatenate_8[0]
dropout (Dropout)	(None, 4, 4, 1024)	0	
average_pooling2d[0]...			
flatten (Flatten)	(None, 16384)	0	dropout[0][0]
dense (Dense)	(None, 3)	49,155	flatten[0][0]

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

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

Non-trainable params: 0 (0.00 B)

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

```

# Compile model
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Early stopping
early_stopping = EarlyStopping(
    monitor='val_accuracy',
    patience=5,
    mode='max',
    restore_best_weights=True
)

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

Epoch 1/30
8/8 _____ 41s 2s/step - accuracy: 0.4250 - loss: 5.8913
- val_accuracy: 0.3125 - val_loss: 1.1134
Epoch 2/30
8/8 _____ 7s 930ms/step - accuracy: 0.3743 - loss:
1.1250 - val_accuracy: 0.3750 - val_loss: 1.0332
Epoch 3/30
8/8 _____ 7s 909ms/step - accuracy: 0.4043 - loss:
1.0434 - val_accuracy: 0.6875 - val_loss: 0.5939
Epoch 4/30
8/8 _____ 7s 904ms/step - accuracy: 0.6535 - loss:
0.7823 - val_accuracy: 0.6875 - val_loss: 0.5692
Epoch 5/30
8/8 _____ 7s 907ms/step - accuracy: 0.6810 - loss:
0.6195 - val_accuracy: 0.6875 - val_loss: 0.5491
Epoch 6/30
8/8 _____ 7s 932ms/step - accuracy: 0.6705 - loss:
0.5391 - val_accuracy: 0.9688 - val_loss: 0.2079
Epoch 7/30
8/8 _____ 8s 927ms/step - accuracy: 0.9391 - loss:
0.6555 - val_accuracy: 0.6875 - val_loss: 4.1603
Epoch 8/30
8/8 _____ 7s 898ms/step - accuracy: 0.6145 - loss:
2.2475 - val_accuracy: 0.6562 - val_loss: 0.5772
Epoch 9/30
8/8 _____ 7s 885ms/step - accuracy: 0.6299 - loss:
0.6222 - val_accuracy: 0.5938 - val_loss: 0.7565

```

Epoch 10/30

8/8 ————— 7s 891ms/step - accuracy: 0.6146 - loss: 0.7133 - val_accuracy: 0.6562 - val_loss: 0.5243

Epoch 11/30

8/8 ————— 7s 932ms/step - accuracy: 0.6407 - loss: 0.5572 - val_accuracy: 0.6875 - val_loss: 0.4977

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

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

```
plt.subplot(1, 2, 1)
```

```
plt.plot(epochs_range, history.history['accuracy'], label='Training Accuracy')
```

```
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation Accuracy')
```

```
plt.legend(loc='lower right')
```

```
plt.title('Training and Validation Accuracy')
```

```
plt.subplot(1, 2, 2)
```

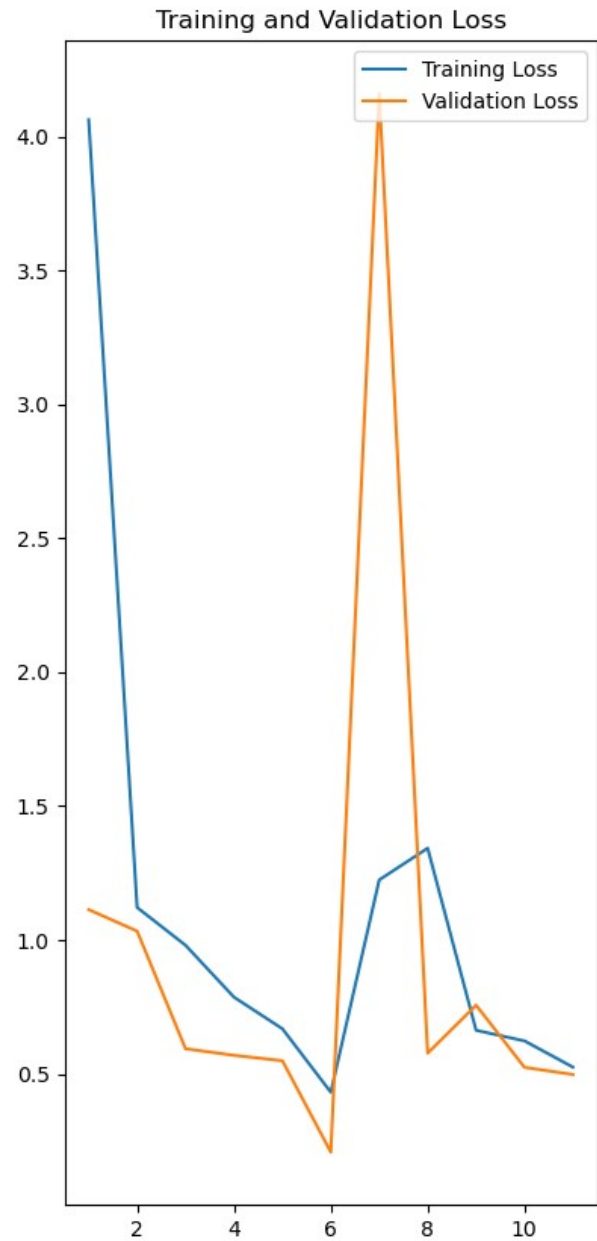
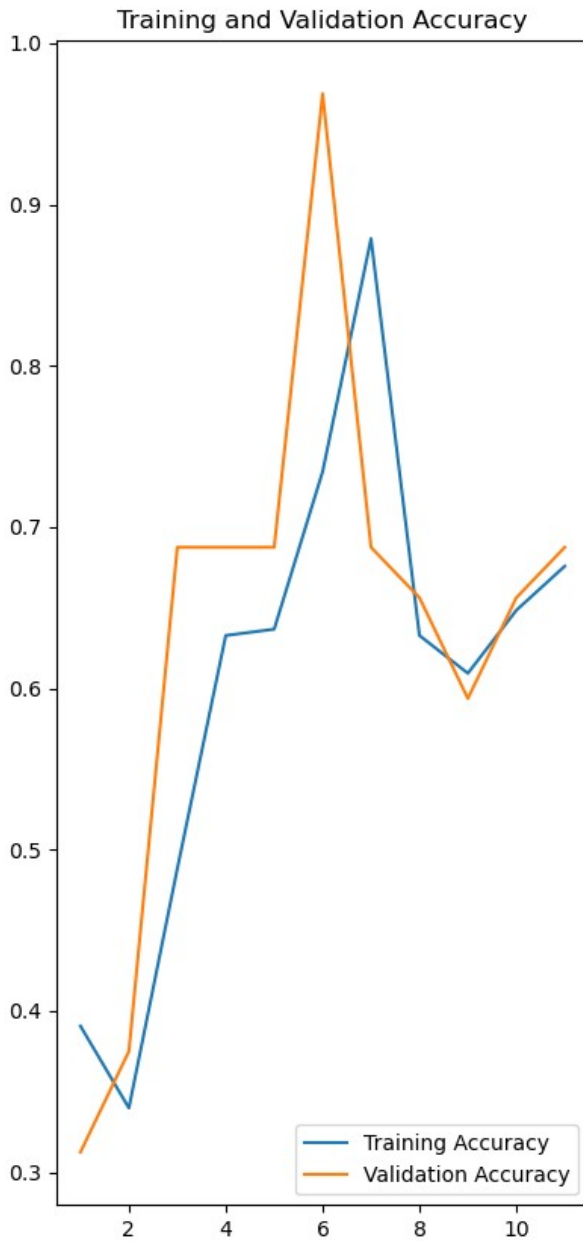
```
plt.plot(epochs_range, history.history['loss'], label='Training Loss')
```

```
plt.plot(epochs_range, history.history['val_loss'], label='Validation Loss')
```

```
plt.legend(loc='upper right')
```

```
plt.title('Training and Validation Loss')
```

```
plt.show()
```



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

```
c:\Users\Lenovo\anaconda3\Lib\site-packages\keras\src\models\
model.py:342: UserWarning: You are saving your model as an HDF5 file
via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  warnings.warn(
```

```
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\Lenovo\Documents\Kuliah - UAJY\Semester
V - 2024-2025\Pembelajaran mesin dan Pembelajaran Mendalam - A\Projek
UAS\googlenet_paprika.h5')
class_names = ['Hijau', 'Kuning', 'Merah']

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

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

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

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

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

# Example usage
result = classify_images(r'C:\Users\Lenovo\Documents\Kuliah - UAJY\
Semester V - 2024-2025\Pembelajaran mesin dan Pembelajaran Mendalam -
A\Projek UAS\data_test\Kuning\Prediksi_7.jpeg',
save_path='kuning.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 ————— 4s 4s/step
Prediksi: Kuning
Confidence: 55.18%
Prediksi: Kuning dengan confidence 55.18%. Gambar asli disimpan di
kuning.jpg.

```

```

import tensorflow as tf
from tensorflow.keras.models import load_model

```

```

import seaborn as sns
import matplotlib.pyplot as plt

# Load test data
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\Lenovo\Documents\Kuliah - UAJY\Semester V - 2024-2025\
Pembelajaran mesin dan Pembelajaran Mendalam - A\Projek UAS\
data_test',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

# Predict model
y_pred = 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)

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

# Compute precision and 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)

# Compute F1 Score
f1_score = 2 * (precision * recall) / (precision + recall)

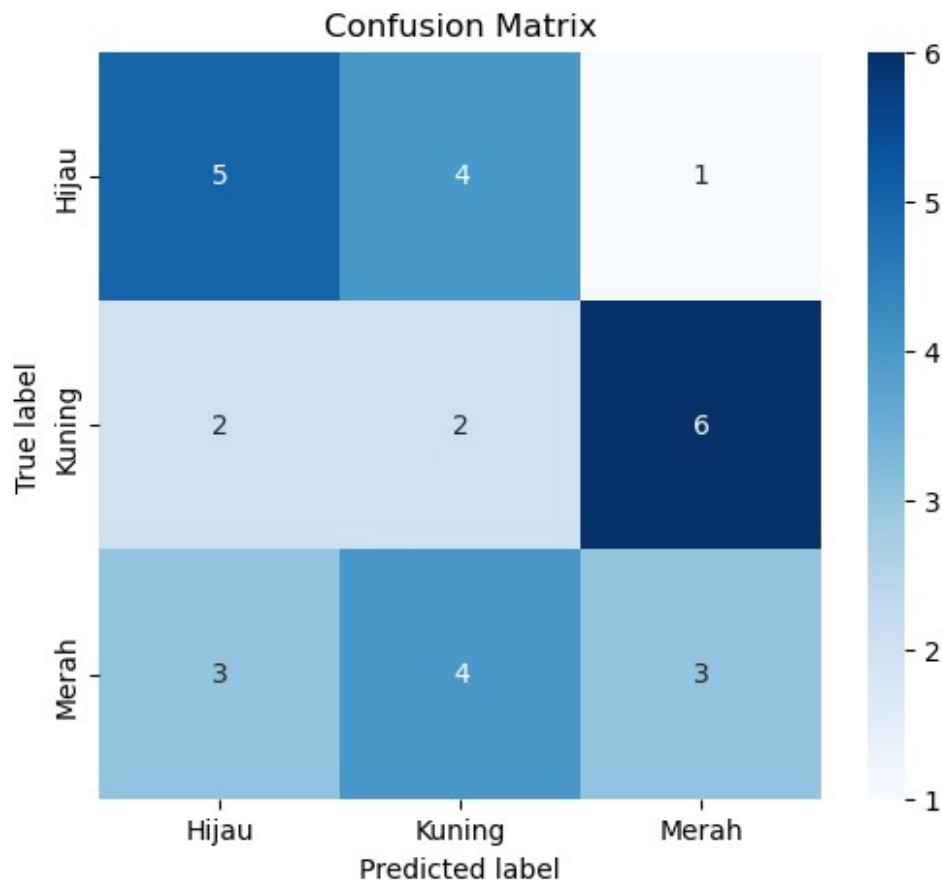
# Visualize Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Hijau", "Kuning", "Merah"],
            yticklabels=["Hijau", "Kuning", "Merah"])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True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

```



```
# Display results
print("Confusion Matrix\n", conf_mat.numpy())
print("Akurasi:\n", accuracy.numpy())
print("Prediksi\n", precision.numpy())
print("F1 Score\n", f1_score.numpy())
```

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



Confusion Matrix

```
[[5 4 1]
 [2 2 6]
 [3 4 3]]
Akurasi:
0.3333333333333333
Prediksi
[0.5 0.2 0.3]
F1 Score
[0.5 0.2 0.3]
```

```
print("Josua Waraney William Lantang / 220712071 / Seaborn/ Perbedaan  
Paprika Merah, Kuning Hijau Menggunakan GoogleNet")
```

Josua Waraney William Lantang / 220712071 / Seaborn/ Perbedaan Paprika Merah, Kuning Hijau Menggunakan GoogleNet

```

import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
from keras.applications import VGG16
from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

img_size = 180
batch_size = 32
random_state = 71
data_dir = "Dataset"

def load_data(directory, img_size, batch_size, random_state):
    dataset = tf.keras.utils.image_dataset_from_directory(
        directory,
        seed=random_state,
        image_size=(img_size, img_size),
        batch_size=batch_size
    )
    return dataset

dataset = load_data(data_dir, img_size, batch_size, random_state)
class_names = dataset.class_names

Found 330 files belonging to 3 classes.

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

visualize_data(dataset, 9)

```

PaprikaMerah



PaprikaMerah



PaprikaMerah



PaprikaHijau



PaprikaHijau



PaprikaMerah



PaprikaMerah



PaprikaHijau



PaprikaMerah



```
val_split = 0.1
test_split = 0.1
total_count = len(dataset)
val_size = int(total_count * val_split)
test_size = int(total_count * test_split)
train_size = total_count - val_size - test_size

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

```

def normalize(image, label):
    return image / 255.0, label

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

def augment_and_normalize(image, label):
    image = data_augmentation(image)
    return normalize(image, label)

train_ds = train_ds.map(augment_and_normalize)
val_ds = val_ds.map(normalize)
test_ds = test_ds.map(normalize)

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        augmented_image = tf.clip_by_value(augmented_images[0] * 255,
0, 255) # Scale back to 0-255
        plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_image.numpy().astype("uint8"))
        plt.axis("off")
plt.show()

```



```
vgg_base = VGG16(weights=None, include_top=False,  
input_shape=(img_size, img_size, 3))
```

```
model = Sequential([  
    vgg_base,  
    Flatten(),  
    Dense(256, activation='relu'),  
    Dropout(0.5),  
    Dense(128, activation='relu'),  
    Dropout(0.5),  
    Dense(len(class_names), activation='softmax')  
])
```

```

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

```

```
model.summary()
```

Model: "sequential_26"

Layer (type) Param #	Output Shape	
vgg16 (Functional) 14,714,688	(None, 5, 5, 512)	
flatten_8 (Flatten) 0	(None, 12800)	
dense_24 (Dense) 3,277,056	(None, 256)	
dropout_16 (Dropout) 0	(None, 256)	
dense_25 (Dense) 32,896	(None, 128)	
dropout_17 (Dropout) 0	(None, 128)	
dense_26 (Dense) 387	(None, 3)	

Total params: 18,025,027 (68.76 MB)

Trainable params: 18,025,027 (68.76 MB)

Non-trainable params: 0 (0.00 B)

```

callbacks = [
    ModelCheckpoint("vgg16_best_model.keras", save_best_only=True,
monitor="val_accuracy", mode="max"),
    EarlyStopping(monitor="val_loss", patience=5,
restore_best_weights=True)
]

```

```

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=25,
    callbacks=callbacks
)

```

Epoch 1/25

```

9/9 _____ 14s 1s/step - accuracy: 0.3714 - loss: 1.0983
- val_accuracy: 0.5625 - val_loss: 1.0972

```

Epoch 2/25

```

9/9 _____ 9s 893ms/step - accuracy: 0.4101 - loss:
1.0973 - val_accuracy: 0.3750 - val_loss: 1.0968

```

Epoch 3/25

```

9/9 _____ 8s 887ms/step - accuracy: 0.3665 - loss:
1.0968 - val_accuracy: 0.5000 - val_loss: 1.0960

```

Epoch 4/25

```

9/9 _____ 9s 1000ms/step - accuracy: 0.5327 - loss:
1.0944 - val_accuracy: 0.7500 - val_loss: 1.0889

```

Epoch 5/25

```

9/9 _____ 9s 1s/step - accuracy: 0.5045 - loss: 1.0900
- val_accuracy: 0.8125 - val_loss: 1.0776

```

Epoch 6/25

```

9/9 _____ 8s 889ms/step - accuracy: 0.6178 - loss:
1.0778 - val_accuracy: 0.7812 - val_loss: 1.0486

```

Epoch 7/25

```

9/9 _____ 8s 885ms/step - accuracy: 0.6529 - loss:
1.0338 - val_accuracy: 0.5938 - val_loss: 0.9586

```

Epoch 8/25

```

9/9 _____ 9s 902ms/step - accuracy: 0.6943 - loss:
0.9233 - val_accuracy: 0.7812 - val_loss: 0.7404

```

Epoch 9/25

```

9/9 _____ 8s 887ms/step - accuracy: 0.6423 - loss:
0.7556 - val_accuracy: 0.7500 - val_loss: 0.5837

```

Epoch 10/25

```

9/9 _____ 10s 1s/step - accuracy: 0.6857 - loss: 0.6532
- val_accuracy: 0.9375 - val_loss: 0.4650

```

Epoch 11/25

```

9/9 _____ 9s 895ms/step - accuracy: 0.6794 - loss:
0.6216 - val_accuracy: 0.9375 - val_loss: 0.4237

```

Epoch 12/25

```

9/9 _____ 8s 896ms/step - accuracy: 0.7250 - loss:
0.6217 - val_accuracy: 0.7188 - val_loss: 0.5419

```



```

Epoch 13/25
9/9 _____ 9s 898ms/step - accuracy: 0.6878 - loss:
0.6322 - val_accuracy: 0.8750 - val_loss: 0.3068
Epoch 14/25
9/9 _____ 9s 1s/step - accuracy: 0.7711 - loss: 0.5165
- val_accuracy: 0.9688 - val_loss: 0.3747
Epoch 15/25
9/9 _____ 9s 920ms/step - accuracy: 0.8577 - loss:
0.4382 - val_accuracy: 0.9688 - val_loss: 0.1936
Epoch 16/25
9/9 _____ 8s 895ms/step - accuracy: 0.8924 - loss:
0.3769 - val_accuracy: 0.9688 - val_loss: 0.0995
Epoch 17/25
9/9 _____ 8s 890ms/step - accuracy: 0.9382 - loss:
0.2553 - val_accuracy: 0.9688 - val_loss: 0.1533
Epoch 18/25
9/9 _____ 9s 1s/step - accuracy: 0.9177 - loss: 0.2366
- val_accuracy: 1.0000 - val_loss: 0.0614
Epoch 19/25
9/9 _____ 8s 887ms/step - accuracy: 0.8992 - loss:
0.2775 - val_accuracy: 0.9688 - val_loss: 0.0568
Epoch 20/25
9/9 _____ 8s 894ms/step - accuracy: 0.9378 - loss:
0.1884 - val_accuracy: 0.8750 - val_loss: 0.3081
Epoch 21/25
9/9 _____ 9s 894ms/step - accuracy: 0.9002 - loss:
0.3023 - val_accuracy: 0.9062 - val_loss: 0.1789
Epoch 22/25
9/9 _____ 8s 885ms/step - accuracy: 0.9352 - loss:
0.2239 - val_accuracy: 0.9688 - val_loss: 0.2132
Epoch 23/25
9/9 _____ 9s 906ms/step - accuracy: 0.9557 - loss:
0.1882 - val_accuracy: 0.9688 - val_loss: 0.0769
Epoch 24/25
9/9 _____ 8s 886ms/step - accuracy: 0.9787 - loss:
0.1385 - val_accuracy: 0.9688 - val_loss: 0.0828

```

```

history_df = pd.DataFrame(history.history)
plt.figure(figsize=(12, 6))

```

```

plt.subplot(1, 2, 1)
plt.plot(history_df['accuracy'], label='Training Accuracy')
plt.plot(history_df['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

```

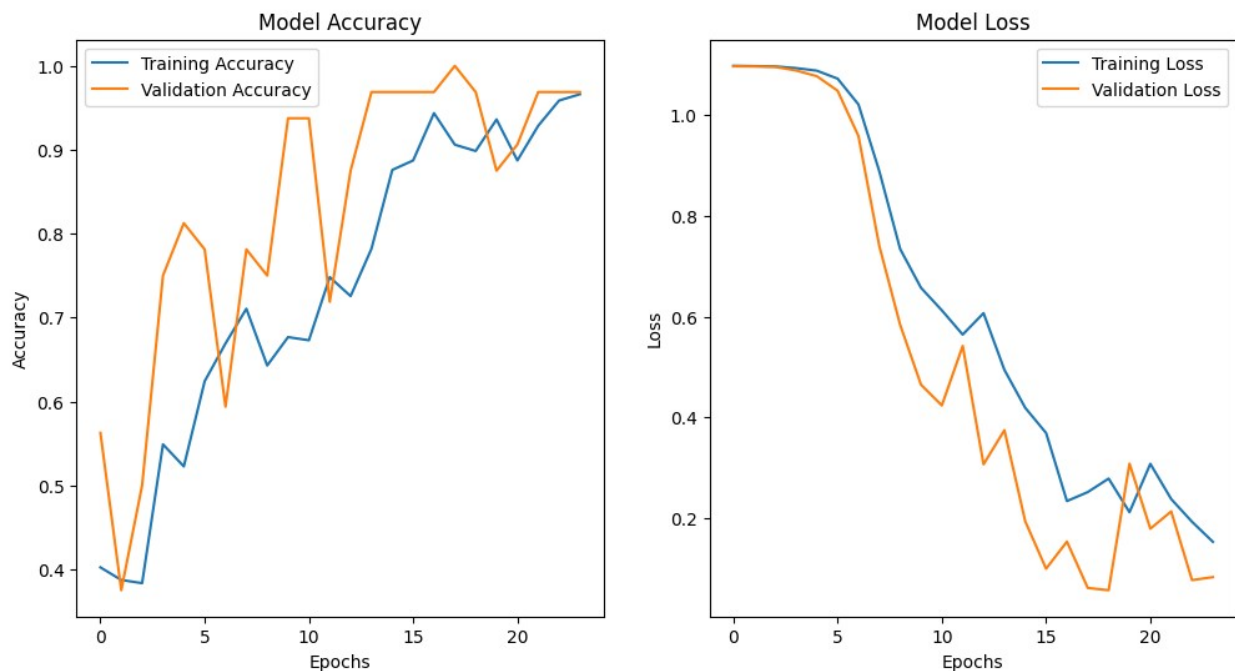
```

plt.subplot(1, 2, 2)
plt.plot(history_df['loss'], label='Training Loss')

```

```
plt.plot(history_df['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



```
def evaluate_model(model, test_ds):
    test_images, test_labels = [], []
    for images, labels in test_ds:
        test_images.append(images.numpy())
        test_labels.append(labels.numpy())

    test_images = np.concatenate(test_images)
    test_labels = np.concatenate(test_labels)

    predictions = np.argmax(model.predict(test_images), axis=1)

    cm = confusion_matrix(test_labels, predictions)

    accuracy = np.trace(cm) / np.sum(cm)
    precision = np.diag(cm) / np.sum(cm, axis=0)
    recall = np.diag(cm) / np.sum(cm, axis=1)
    f1_score = 2 * (precision * recall) / (precision + recall)

    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=class_names, yticklabels=class_names)
```

```

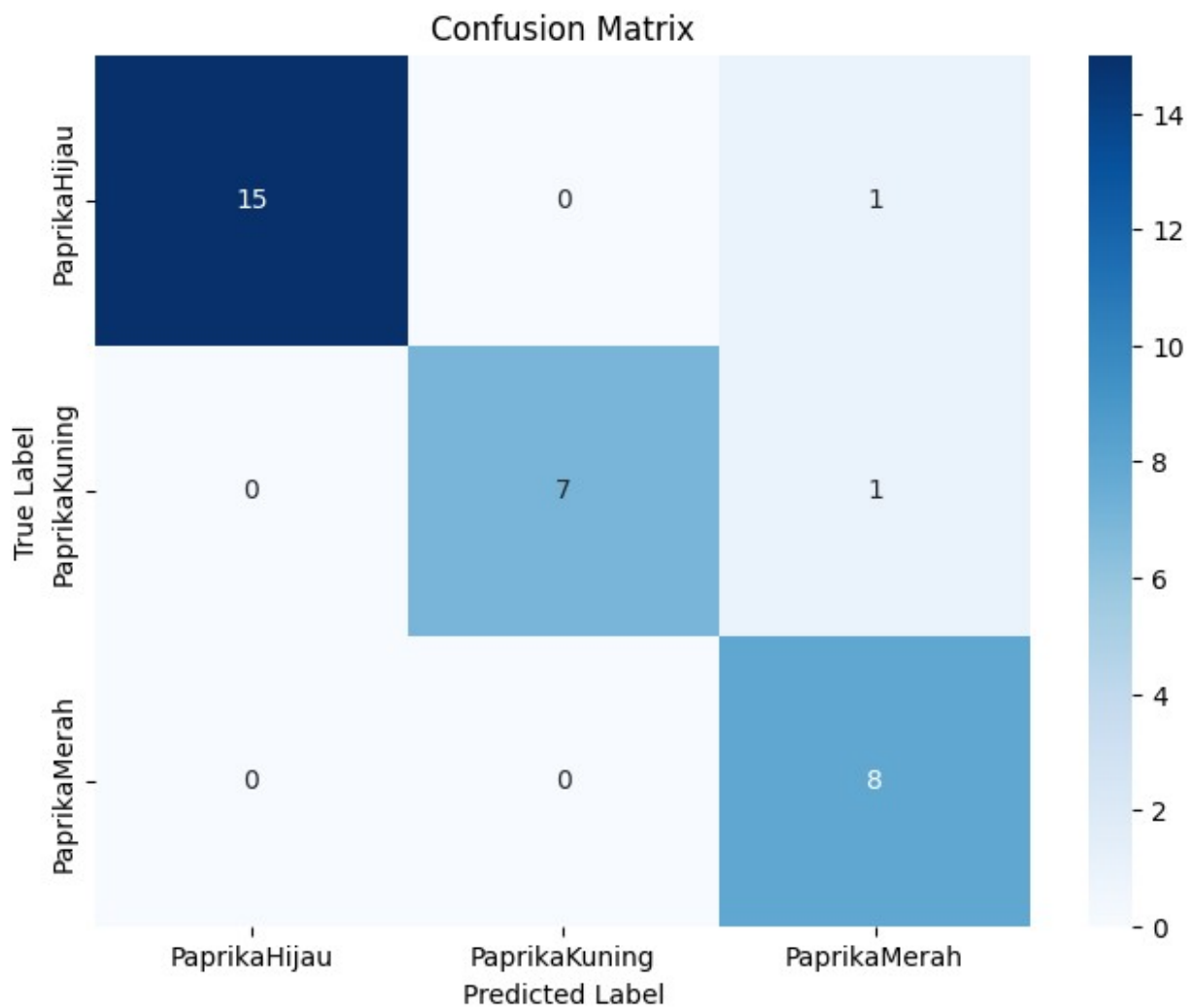
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

print("Confusion Matrix:")
print(cm)
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1_score}")

```

```
evaluate_model(model, test_ds)
```

1/1 ————— 0s 497ms/step



Confusion Matrix:

```
[[15  0  1]
 [ 0  7  1]
 [ 0  0  8]]
```

Accuracy: 0.9375

Precision: [1. 1. 0.8]

Recall: [0.9375 0.875 1.]

F1 Score: [0.96774194 0.93333333 0.88888889]

```
model.save("BestModel_VGG-16_Seaborn.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:
        input_image = tf.keras.utils.load_img(image_path,
        target_size=(img_size, img_size))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)

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

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

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

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

result = classify_images(r'.jpg', save_path='paprika.jpg')
print(result)
```

1/1 ————— 0s 64ms/step

Prediksi: PaprikaHijau

Confidence: 57.61%

Prediksi: PaprikaHijau dengan confidence 57.61%. Gambar asli disimpan di paprika.jpg.

```
print("Eliandani Andreskia Setiawan / 220711965 / Seaborn / Perbedaan
Paprika Merah, Kuning, Hijau Menggunaakn VGG-16")
```

Eliandani Andreskia Setiawan / 220711965 / Seaborn / Perbedaan Paprika Merah, Kuning, Hijau Menggunakan VGG-16

30l2mi5z0

December 19, 2024

```
[171]: 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
```

```
[172]: count = 0 # digunakan untuk menghitung jumlah gambar
base_path = r'D:\UAJY\UAJY_sem_5\ML\UAS\data_paprika'
dirs = os.listdir(base_path)

for dir in dirs:
    # Gunakan os.path.join untuk menggabungkan path dengan benar
    full_path = os.path.join(base_path, dir)
    files = list(os.listdir(full_path))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
PaprikaHijau Folder has 110 Images
PaprikaKuning Folder has 110 Images
PaprikaMerah Folder has 110 Images
Images Folder has 330 Images
```

```
[173]: base_dir = r'data_paprika'
img_size = 180
batch = 32
```

```
[174]: train_ds = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    validation_split=0.2, # Pisahkan 20% data
    subset="training",    # Ambil 80% untuk training
    seed=123,
    image_size=(img_size, img_size),
    batch_size=batch,
```

```
)
```

Found 330 files belonging to 3 classes.
Using 264 files for training.

```
[175]: val_test_ds = tf.keras.utils.image_dataset_from_directory(  
        base_dir,  
        validation_split=0.2,  
        subset="validation",  
        seed=123,  
        image_size=(img_size, img_size),  
        batch_size=batch,  
    )
```

Found 330 files belonging to 3 classes.
Using 66 files for validation.

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

Class Names: ['PaprikaHijau', 'PaprikaKuning', 'PaprikaMerah']

```
[177]: val_size = len(val_test_ds) // 2  
val_ds = val_test_ds.take(val_size)  
test_ds = val_test_ds.skip(val_size)  
  
print("Total Images:", len(dataset))  
print("Train Images:", len(train_ds))  
print("Validation Images:", len(val_ds))  
print("Test Images:", len(test_ds))
```

Total Images: 11
Train Images: 9
Validation Images: 1
Test Images: 2

```
[178]: train_ds = dataset.take(train_count)  
val_ds = dataset.skip(train_count)
```

```
[179]: 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')
```

PaprikaHijau



PaprikaKuning



PaprikaKuning



PaprikaHijau



PaprikaHijau



PaprikaMerah



PaprikaHijau



PaprikaHijau



PaprikaMerah



```
[180]: 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)


```
[181]: AUTOTUNE = tf.data.AUTOTUNE
```

```
[182]: train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

```
[183]: val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

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

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

```
    super().__init__(**kwargs)
```

```
[185]: i = 0
plt.figure(figsize=(10,10))

for images, labels in train_ds.take(3):
    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')
```



```
[186]: #import library yang dibutuhkan
from tensorflow.keras.applications import MobileNet #digunakan untuk
    ↳ memanfaatkan model yang sudah dilatih sebelumnya untuk pengenalan gambar
from tensorflow.keras.models import Model #digunakan untuk membuat dan
    ↳ mengonfigurasi arsitektur model

#membuat model dengan bobot yang telah dilatih sebelumnya
#include_top=False berarti tidak menggunakan lapisan klasifikasi dari mobilenet
    ↳ hanya bagian ekstraksi fitur
base_model = MobileNet(include_top=False, input_shape=(img_size, img_size, 3))

#membuka (unfreeze beberapa lapisan untuk proses fine tuning)
base_model.trainable = True #seluruh model bisa dilatih
```

```

fine_tune_at = len(base_model.layers) // 2 #menentukan bahwa setengah lapisan
↳ terakhir akan di unfreeze
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False #mengunci (freeze) lapisan pertama hingga setengah
↳ bagian pertama agar tidak dilatih kembali

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')
])

#membuat model akhir dengan lapisan tambahan
###Terdapat code yang hilang disini! lihat modul untuk menemukanya

```

C:\Users\ACER\AppData\Local\Temp\ipykernel_13632\2025012950.py:7: 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))

```

[187]: from tensorflow.keras.optimizers import Adam #untuk mengoptimalkan proses
↳ pelatihan model

#mengkompilasi model dengan optimizer, loss function, dan metrics
model.compile(
    optimizer=Adam(learning_rate=1e-4), #menggunakan optimizer Adam dengan
↳ learning rate 0.0001
    loss='sparse_categorical_crossentropy', #untuk klasifikasi multi-kelas
    metrics=['accuracy'] #akurasi digunakan sebagai metrik evaluasi
)

```

```

[188]: #menampilkan ringkasan dari model
model.summary()

```

Model: "sequential_15"

Layer (type)	Output Shape	Param #
sequential_14 (Sequential)	(None, 180, 180, 3)	0
rescaling_6 (Rescaling)	(None, 180, 180, 3)	0

mobilenet_1.00_224 (Functional)	(None, 5, 5, 1024)	3,228,864
global_average_pooling2d_7 (GlobalAveragePooling2D)	(None, 1024)	0
dense_15 (Dense)	(None, 128)	131,200
dropout_8 (Dropout)	(None, 128)	0
dense_16 (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)

```
[189]: #early stopping digunakan untuk menghentikan pelatihan lebih awal jika model
        ↳ tidak ada peningkatan
from tensorflow.keras.callbacks import EarlyStopping

#Ada fungsi early stopping disini, jangan keskip tuan :D
early_stopping = EarlyStopping(monitor='val_accuracy',
                                patience=3,
                                mode='max')

#melatih model menggunakan data latih dan validasi dengan early stopping
history= model.fit(train_ds, #data pelatihan yang telah disiapkan
                    epochs=30, # jumlah maksimal epoch
                    validation_data=val_ds, #data validasi untuk mengevaluasi
                    ↳model pada setiap epoch
                    callbacks=[early_stopping]) #menambahkan early stopping ke
                    ↳dalam callback untuk pelatihan
```

Epoch 1/30

10/10 14s 511ms/step -

accuracy: 0.4416 - loss: 1.2186 - val_accuracy: 0.3000 - val_loss: 1.5862

Epoch 2/30

10/10 4s 420ms/step -

accuracy: 0.7443 - loss: 0.6298 - val_accuracy: 0.4000 - val_loss: 1.3009

Epoch 3/30

10/10 5s 501ms/step -

accuracy: 0.8306 - loss: 0.4017 - val_accuracy: 0.5000 - val_loss: 1.0949

Epoch 4/30

```

10/10          4s 411ms/step -
accuracy: 0.8866 - loss: 0.2812 - val_accuracy: 0.5000 - val_loss: 0.9794
Epoch 5/30
10/10          4s 401ms/step -
accuracy: 0.9465 - loss: 0.1775 - val_accuracy: 0.8000 - val_loss: 0.6003
Epoch 6/30
10/10          4s 402ms/step -
accuracy: 0.9804 - loss: 0.0908 - val_accuracy: 0.9000 - val_loss: 0.3741
Epoch 7/30
10/10          4s 414ms/step -
accuracy: 0.9874 - loss: 0.0764 - val_accuracy: 0.9000 - val_loss: 0.2571
Epoch 8/30
10/10          4s 402ms/step -
accuracy: 0.9739 - loss: 0.0869 - val_accuracy: 0.9000 - val_loss: 0.1717
Epoch 9/30
10/10          4s 406ms/step -
accuracy: 0.9893 - loss: 0.0614 - val_accuracy: 0.9000 - val_loss: 0.1277

```

```

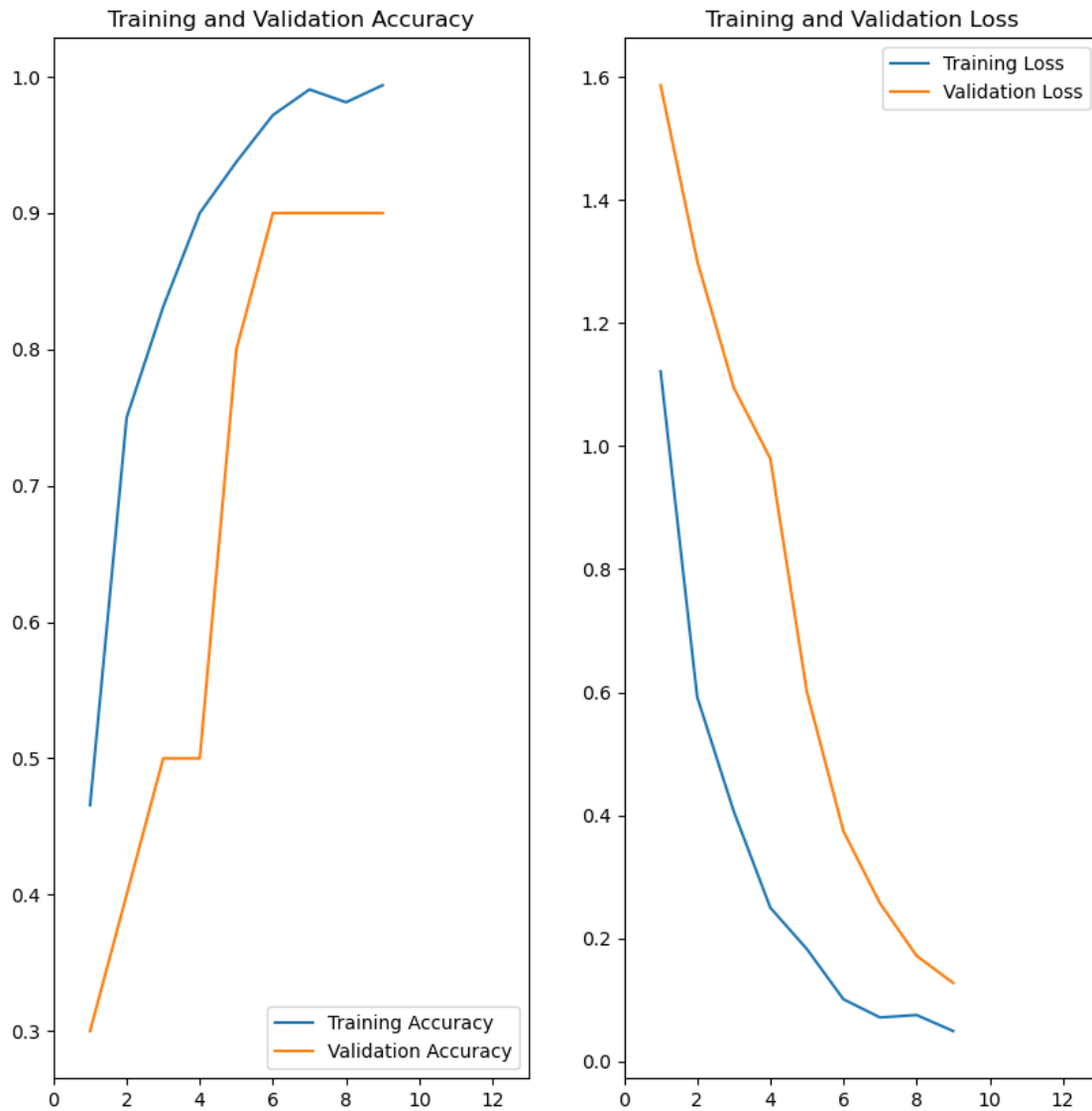
[190]: ephocs_range = range(1, len(history.history['loss']) + 1)

plt.figure(figsize=(10, 10)) #membuat figure dengan ukuran 10x10 untuk
    ↪menampilkan 2 grafik (Training and Validation Accuracy dan Loss)

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()

```



```
[191]: model.save('BestModel_MobilNetCNN_Seaborn.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')`.

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

```

model = load_model(r'BestModel_MobilNetCNN_Seaborn.h5')
class_names = ['PaprikaHijau', 'PaprikaKuning', 'PaprikaMerah' ]

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)) #membuat gambar dari path dan mnegubah ukurannya menjadi 180x180 pixel
        input_image_array = tf.keras.utils.img_to_array(input_image) #mengubah
↪gambar jadi array numpy agar bisa di proses model
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)
↪#menambahkan dimensi batch agar sesuai dengan input model
                                                                    #dimensi
↪menjadi (1, 180, 180, 3)

        #melakukan prediksi
        predictions = model.predict(input_image_exp_dim) #melakukan prediksi
↪pada gambar yang telah diproses
        result = tf.nn.softmax(predictions[0]) #menghitung hasil prediksi
↪menggunakan softmax untuk mendapatkan probabilitas tiap kelas
        class_idx = np.argmax(result) #menemukan indeks kelas dengan
↪probabilitas tertinggi
        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:
↳\UAJY\UAJY_sem_5\ML\UAS\data_paprika\PaprikaHijau\Green_42.jpg" ,
↳save_path='hijau.jpg')
print(result)
###Terdapat code yang hilang disini! lihat modul untuk menemukanya

```

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 712ms/step

Prediksi: PaprikaHijau

Confidence: 56.10%

Prediksi: PaprikaHijau dengan confidence 56.10%. Gambar asli disimpan di hijau.jpg.

```

[193]: import tensorflow as tf
from tensorflow.keras.models import load_model
import seaborn as sns
import matplotlib.pyplot as plt

#memuat model yang telah dilatih sebelumnya
mobileNet_model = load_model(r'BestModel_MobilNetCNN_Seaborn.h5')#gunakan path
↳masing masing ya

#memuat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'data_paprika', #direktori data uji
    labels='inferred', #label otomatis dari subfolder yang ada
    label_mode='categorical', #menghasilkan label dalam bentuk one-hot encoding
    batch_size=32, #ukuran batch untuk pemrosesan
    image_size=(180, 180) #ukuran gambar yang akan diproses
)

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

#ekstrak label sebenarnya dari test_data dan konversi ke bentuk indeks kelas
true_labels = [] #menyimpan label asli dalam bentuk indeks
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy()) #konversi one-hot ke
↳indeks kelas
true_labels = tf.convert_to_tensor(true_labels) #mengkonversi list ke tensor
↳untuk perhitungan

#membuat confusion matrix untuk evaluasi

```



```

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

#menghitung akurasi berdasarkan confusion matrix
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) / tf.
    ↳reduce_sum(conf_mat)

#menghitung presisi dan recall dari confusion matrix
precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat, axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat, axis=1)

#menghitung F1 Score
f1_score = 2 * (precision * recall) / (precision + recall)

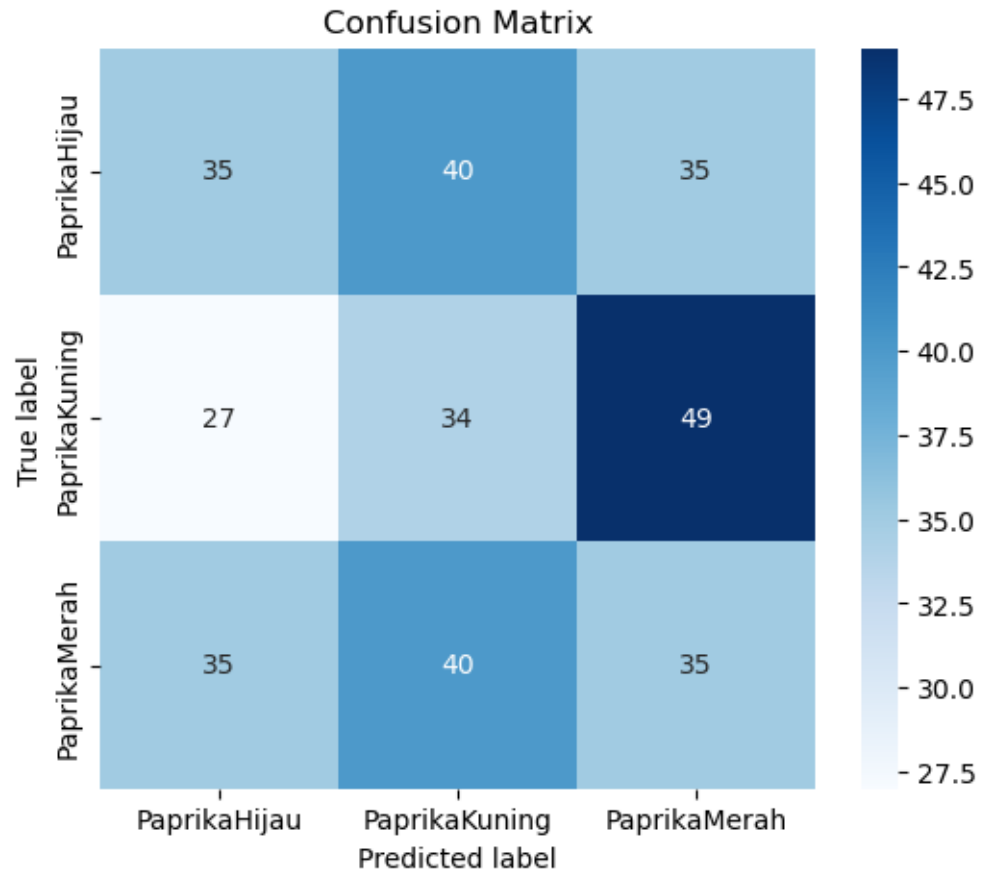
#visualisasi Confusion Matrix
plt.figure(figsize=(6, 5)) #mengatur ukuran gambar
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues', #annot=True,
    ↳untuk menampilkan angka di dalam setiap sel matriks
                                                    #fmt='d' untuk
    ↳menampilkan bilangan bulat tanpa desimal
            xticklabels=["PaprikaHijau", "PaprikaKuning", "PaprikaMerah"],
    ↳yticklabels=["PaprikaHijau", "PaprikaKuning", "PaprikaMerah"])
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 330 files belonging to 3 classes.
 11/11 4s 269ms/step



Confusion Matrix:

```
[[35 40 35]
```

```
[27 34 49]
```

```
[35 40 35]]
```

Akurasi: 0.3151515151515151

Presisi: [0.36082474 0.29824561 0.29411765]

Recall: [0.31818182 0.30909091 0.31818182]

F1 Score: [0.33816425 0.30357143 0.30567686]

```
print("Veronica Regina Mambu / 220711948/ Seaborn/ Perbedaan Paprika Merah, Kuning Hijau  
Menggunaakn MobileNet")
```

```
import tensorflow as tf
import numpy as np
import random
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout, BatchNormalization
from tensorflow.keras.regularizers import l2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

```
seed_value = 71
tf.random.set_seed(seed_value)
np.random.seed(seed_value)
random.seed(seed_value)
```

2024-12-19 20:19:41.453396: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.

2024-12-19 20:19:41.472418: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered

WARNING: All log messages before absl::InitializeLog() is called are written to STDERR

E0000 00:00:1734614381.497586 2685741 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered

E0000 00:00:1734614381.503886 2685741 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered

2024-12-19 20:19:41.527624: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.

To enable the following instructions: AVX2 AVX512F AVX512_VNNI AVX512_BF16 AVX512_FP16 AVX_VNNI AMX_TILE AMX_INT8 AMX_BF16 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.

```
def load_data(directory, img_size=(224, 224), batch_size=32):
    dataset = tf.keras.utils.image_dataset_from_directory(
        directory,
        image_size=img_size,
        batch_size=batch_size,
```

```

        shuffle=True
    )
    return dataset

data_directory = "Dataset UAS"
dataset = load_data(data_directory)

class_names = dataset.class_names
print(f"Class names: {class_names}")

Found 330 files belonging to 3 classes.
Class names: ['PaprikaHijauFix', 'PaprikaKuningFix',
 'PaprikaMerahFix']

W0000 00:00:1734614384.114357 2685741 gpu_device.cc:2344] Cannot
dlopen some GPU libraries. Please make sure the missing libraries
mentioned above are installed properly if you would like to use GPU.
Follow the guide at https://www.tensorflow.org/install/gpu for how to
download and setup the required libraries for your platform.
Skipping registering GPU devices...

def visualize_data(dataset, num_images=9):
    label_map = {
        'PaprikaHijauFix': 'PaprikaHijau',
        'PaprikaKuningFix': 'PaprikaKuning',
        'PaprikaMerahFix': 'PaprikaMerah'
    }

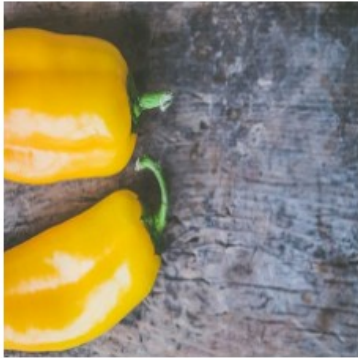
    plt.figure(figsize=(10, 10))
    for images, labels in dataset.take(1):
        for i in range(num_images):
            plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
            # Map label index to class name and custom label
            label_name = class_names[labels[i]]
            custom_label = label_map.get(label_name, "Unknown")
            plt.title(custom_label)
            plt.axis("off")
    plt.show()

visualize_data(dataset)

2024-12-19 20:19:44.585828: I
tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is
aborting with status: OUT_OF_RANGE: End of sequence

```

PaprikaKuning



PaprikaHijau



PaprikaHijau



PaprikaHijau



PaprikaHijau



PaprikaKuning



PaprikaKuning



PaprikaHijau



PaprikaHijau



```
augmentation = ImageDataGenerator(  
    rotation_range=30,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True,  
    fill_mode='nearest'  
)
```

```
def normalize_data(dataset):
    return dataset.map(lambda x, y: (tf.cast(x, tf.float32) / 255.0,
y))
```

```
def split_dataset(dataset, train_ratio=0.8, val_ratio=0.1):
```

```
    dataset_size = len(dataset)
    train_size = int(train_ratio * dataset_size)
    val_size = int(val_ratio * dataset_size)

    train_dataset = dataset.take(train_size)
    val_test_dataset = dataset.skip(train_size)
    val_dataset = val_test_dataset.take(val_size)
    test_dataset = val_test_dataset.skip(val_size)
```

```
    return train_dataset, val_dataset, test_dataset
```

```
train_ds, val_ds, test_ds = split_dataset(dataset)
```

```
train_ds = normalize_data(train_ds)
val_ds = normalize_data(val_ds)
test_ds = normalize_data(test_ds)
```

```
print(f"Train size: {len(train_ds)}, Validation size: {len(val_ds)},
Test size: {len(test_ds)}")
```

```
Train size: 8, Validation size: 1, Test size: 2
```

```
def build_alexnet(input_shape=(224, 224, 3), num_classes=3):
```

```
    model = Sequential([
        Conv2D(96, (11, 11), strides=4, activation='relu',
input_shape=input_shape,
        kernel_regularizer=l2(0.01)),
        MaxPooling2D((3, 3), strides=2),
        BatchNormalization(),
        Dropout(0.3),

        Conv2D(256, (5, 5), activation='relu', padding='same',
kernel_regularizer=l2(0.01)),
        MaxPooling2D((3, 3), strides=2),
        BatchNormalization(),
        Dropout(0.3),

        Conv2D(384, (3, 3), activation='relu', padding='same'),
        Conv2D(384, (3, 3), activation='relu', padding='same'),
        Conv2D(256, (3, 3), activation='relu', padding='same'),
        MaxPooling2D((3, 3), strides=2),
        BatchNormalization(),
        Dropout(0.3),
```

```

        Flatten(),
        Dense(4096, activation='relu', kernel_regularizer=l2(0.01)),
        Dropout(0.5),
        Dense(4096, activation='relu', kernel_regularizer=l2(0.01)),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

```

```

model = build_alexnet()
model.summary()

```

/opt/tljh/user/envs/dlthf/lib/python3.10/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: "sequential"

Layer (type) Param #	Output Shape	
conv2d (Conv2D) 34,944	(None, 54, 54, 96)	
max_pooling2d (MaxPooling2D) 0	(None, 26, 26, 96)	
batch_normalization 384 (BatchNormalization)	(None, 26, 26, 96)	
dropout (Dropout) 0	(None, 26, 26, 96)	
conv2d_1 (Conv2D)	(None, 26, 26, 256)	

614,656		
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 256)	
0		
batch_normalization_1	(None, 12, 12, 256)	
1,024		
(BatchNormalization)		
dropout_1 (Dropout)	(None, 12, 12, 256)	
0		
conv2d_2 (Conv2D)	(None, 12, 12, 384)	
885,120		
conv2d_3 (Conv2D)	(None, 12, 12, 384)	
1,327,488		
conv2d_4 (Conv2D)	(None, 12, 12, 256)	
884,992		
max_pooling2d_2 (MaxPooling2D)	(None, 5, 5, 256)	
0		
batch_normalization_2	(None, 5, 5, 256)	
1,024		
(BatchNormalization)		
dropout_2 (Dropout)	(None, 5, 5, 256)	
0		
flatten (Flatten)	(None, 6400)	
0		
dense (Dense)	(None, 4096)	
26,218,496		

0	dropout_3 (Dropout)	(None, 4096)
16,781,312	dense_1 (Dense)	(None, 4096)
0	dropout_4 (Dropout)	(None, 4096)
12,291	dense_2 (Dense)	(None, 3)

Total params: 46,761,731 (178.38 MB)

Trainable params: 46,760,515 (178.38 MB)

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

```

from tensorflow.keras.callbacks import ReduceLROnPlateau,
EarlyStopping

lr_scheduler = ReduceLROnPlateau(
    monitor='val_loss', factor=0.5, patience=5, verbose=1, min_lr=1e-6
)

early_stopping = EarlyStopping(
    monitor='val_loss', patience=10, restore_best_weights=True
)

def train_model(model, train_ds, val_ds, epochs=20):
    history = model.fit(
        train_ds,
        validation_data=val_ds,
        epochs=epochs,
        callbacks=[lr_scheduler, early_stopping]
    )

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

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

```

```

plt.title('Accuracy')

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

plt.show()
return history

```

```
history = train_model(model, train_ds, val_ds)
```

Epoch 1/20

```
8/8 ━━━━━━━━━━━ 6s 284ms/step - accuracy: 0.4834 - loss:
98.8750 - val_accuracy: 0.3750 - val_loss: 128.4318 - learning_rate:
0.0010
```

Epoch 2/20

```
8/8 ━━━━━━━━━━━ 2s 237ms/step - accuracy: 0.9114 - loss:
83.8901 - val_accuracy: 0.3125 - val_loss: 96.0019 - learning_rate:
0.0010
```

Epoch 3/20

```
8/8 ━━━━━━━━━━━ 2s 238ms/step - accuracy: 0.9370 - loss:
73.8499 - val_accuracy: 0.5625 - val_loss: 126.1317 - learning_rate:
0.0010
```

Epoch 4/20

```
8/8 ━━━━━━━━━━━ 2s 234ms/step - accuracy: 0.9097 - loss:
65.1119 - val_accuracy: 0.3750 - val_loss: 94.7901 - learning_rate:
0.0010
```

Epoch 5/20

```
8/8 ━━━━━━━━━━━ 2s 235ms/step - accuracy: 0.9724 - loss:
55.8687 - val_accuracy: 0.6562 - val_loss: 59.6748 - learning_rate:
0.0010
```

Epoch 6/20

```
8/8 ━━━━━━━━━━━ 2s 241ms/step - accuracy: 0.9741 - loss:
48.1750 - val_accuracy: 0.9375 - val_loss: 42.2939 - learning_rate:
0.0010
```

Epoch 7/20

```
8/8 ━━━━━━━━━━━ 2s 241ms/step - accuracy: 0.9765 - loss:
40.5259 - val_accuracy: 0.9688 - val_loss: 35.7874 - learning_rate:
0.0010
```

Epoch 8/20

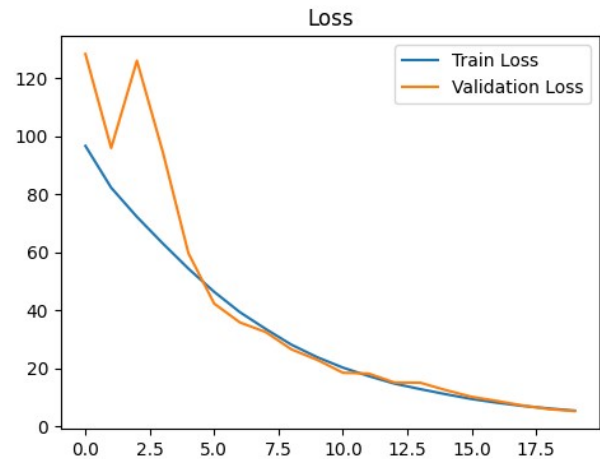
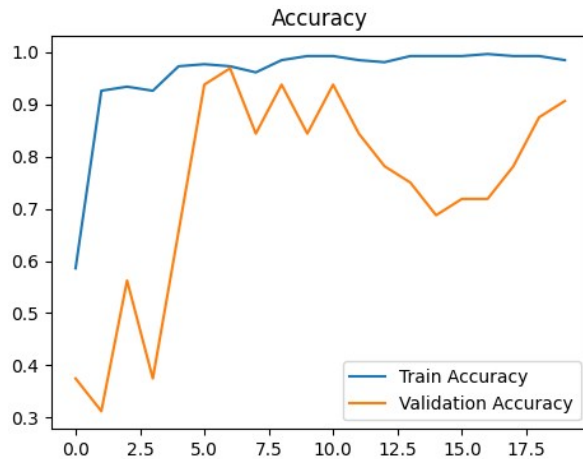
```
8/8 ━━━━━━━━━━━ 2s 246ms/step - accuracy: 0.9602 - loss:
34.5616 - val_accuracy: 0.8438 - val_loss: 32.4782 - learning_rate:
0.0010
```

Epoch 9/20

```
8/8 ━━━━━━━━━━━ 2s 223ms/step - accuracy: 0.9940 - loss:
28.9503 - val_accuracy: 0.9375 - val_loss: 26.4997 - learning_rate:
0.0010
```

Epoch 10/20

8/8 ————— 2s 210ms/step - accuracy: 0.9901 - loss:
24.5835 - val_accuracy: 0.8438 - val_loss: 22.8815 - learning_rate:
0.0010
Epoch 11/20
8/8 ————— 2s 222ms/step - accuracy: 0.9907 - loss:
20.8647 - val_accuracy: 0.9375 - val_loss: 18.4345 - learning_rate:
0.0010
Epoch 12/20
8/8 ————— 2s 210ms/step - accuracy: 0.9889 - loss:
17.6992 - val_accuracy: 0.8438 - val_loss: 18.1380 - learning_rate:
0.0010
Epoch 13/20
8/8 ————— 2s 210ms/step - accuracy: 0.9904 - loss:
15.0889 - val_accuracy: 0.7812 - val_loss: 15.0597 - learning_rate:
0.0010
Epoch 14/20
8/8 ————— 2s 212ms/step - accuracy: 0.9872 - loss:
13.3028 - val_accuracy: 0.7500 - val_loss: 14.9901 - learning_rate:
0.0010
Epoch 15/20
8/8 ————— 2s 207ms/step - accuracy: 0.9889 - loss:
11.4017 - val_accuracy: 0.6875 - val_loss: 12.4215 - learning_rate:
0.0010
Epoch 16/20
8/8 ————— 2s 204ms/step - accuracy: 0.9910 - loss:
9.6262 - val_accuracy: 0.7188 - val_loss: 10.1211 - learning_rate:
0.0010
Epoch 17/20
8/8 ————— 2s 203ms/step - accuracy: 0.9974 - loss:
8.2893 - val_accuracy: 0.7188 - val_loss: 8.6541 - learning_rate:
0.0010
Epoch 18/20
8/8 ————— 2s 205ms/step - accuracy: 0.9967 - loss:
7.1420 - val_accuracy: 0.7812 - val_loss: 7.1087 - learning_rate:
0.0010
Epoch 19/20
8/8 ————— 2s 203ms/step - accuracy: 0.9945 - loss:
6.2593 - val_accuracy: 0.8750 - val_loss: 5.8633 - learning_rate:
0.0010
Epoch 20/20
8/8 ————— 2s 209ms/step - accuracy: 0.9829 - loss:
5.4749 - val_accuracy: 0.9062 - val_loss: 5.1831 - learning_rate:
0.0010



```
def evaluate_model(model, test_ds):

    test_loss, test_acc = model.evaluate(test_ds)
    print(f"Test Accuracy: {test_acc:.2f}")

evaluate_model(model, test_ds)

2/2 ————— 0s 32ms/step - accuracy: 0.9737 - loss:
5.0376
Test Accuracy: 0.98

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

def plot_confusion_matrix(model, test_ds):
    y_true = []
    y_pred = []

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

    cm = confusion_matrix(y_true, y_pred)

    disp = ConfusionMatrixDisplay(
        confusion_matrix=cm, display_labels=class_names
    )
    disp.plot(cmap='viridis', xticks_rotation='vertical')
    plt.title("Confusion Matrix")
    plt.show()

evaluate_model(model, test_ds)

plot_confusion_matrix(model, test_ds)
```

2/2 _____ 0s 25ms/step - accuracy: 0.9420 - loss:

5.3302

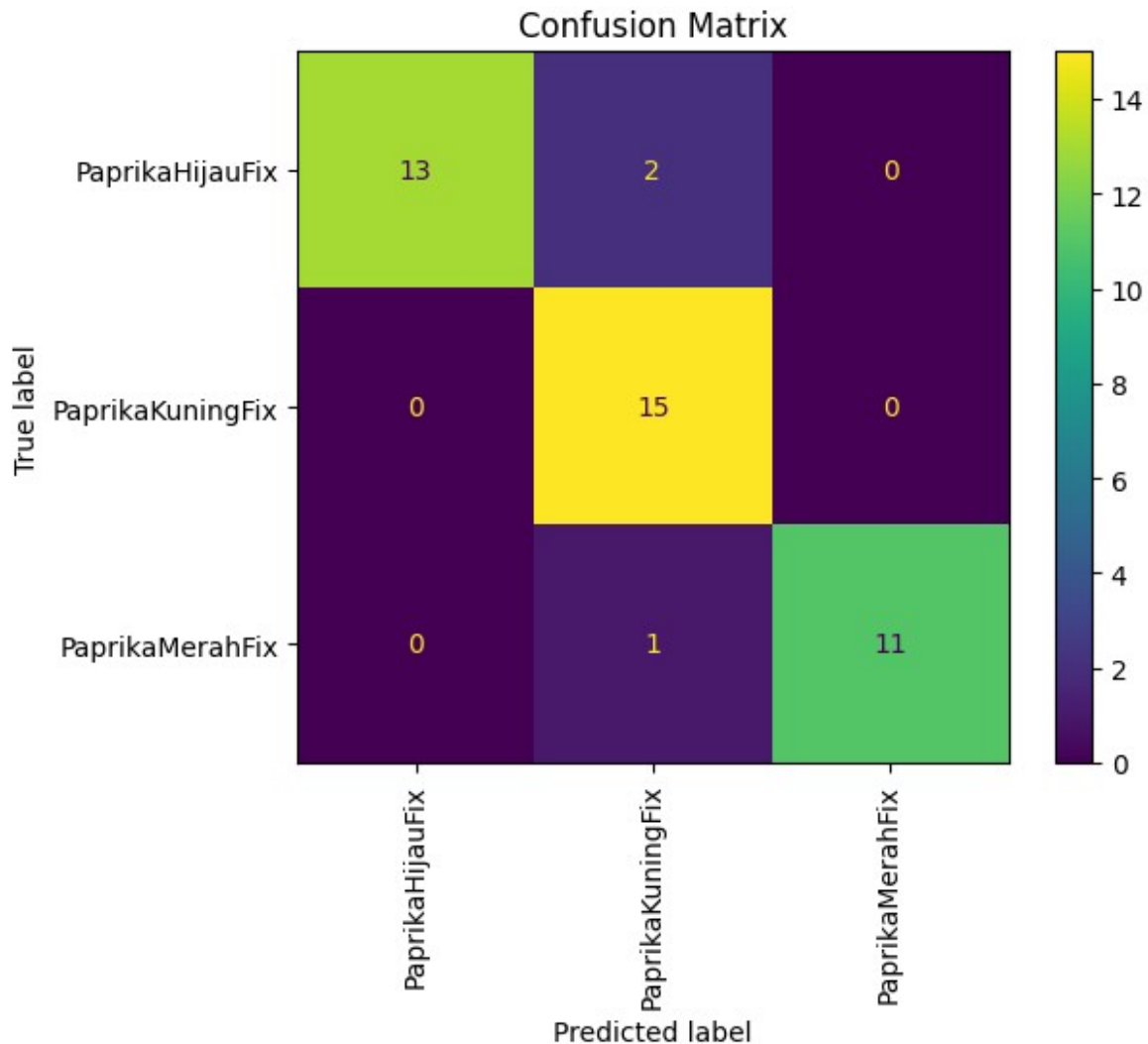
Test Accuracy: 0.93

1/1 _____ 0s 189ms/step

1/1 _____ 0s 153ms/step

2024-12-19 20:20:26.239342: I

tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence



```
model.save("BestModel_AlexNet_Seaborn.h5")
```

```
print("Model saved as alexnet_paprika.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')`.

Model saved as alexnet_paprika.h5

```
from tensorflow.keras.models import load_model
```

```
model_path = "BestModel_AlexNet_Seaborn.h5"  
loaded_model = load_model(model_path)  
print("Model loaded successfully.")
```

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.

Model loaded successfully.

```
from sklearn.metrics import classification_report, confusion_matrix,  
ConfusionMatrixDisplay  
import matplotlib.pyplot as plt
```

```
def evaluate_entire_dataset(model, datasets, dataset_names):
```

```
    for ds, name in zip(datasets, dataset_names):  
        print(f"\nEvaluating on {name} Dataset:")
```

```
        loss, accuracy = model.evaluate(ds)  
        print(f"{name} Accuracy: {accuracy:.2f}")  
        print(f"{name} Loss: {loss:.2f}")
```

```
        y_true = []  
        y_pred = []
```

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

```
        print(f"\n{name} Classification Report:")  
        print(classification_report(y_true, y_pred,  
target_names=class_names))
```


```
        cm = confusion_matrix(y_true, y_pred)  
        disp = ConfusionMatrixDisplay(  
            confusion_matrix=cm, display_labels=class_names  
        )  
        disp.plot(cmap='viridis', xticks_rotation='vertical')  
        plt.title(f"{name} Confusion Matrix")  
        plt.show()
```

```

evaluate_entire_dataset(
    model=loaded_model,
    datasets=[train_ds, val_ds, test_ds],
    dataset_names=["Train", "Validation", "Test"]
)



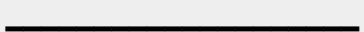
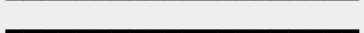
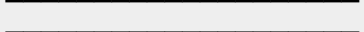



```

Evaluating on Train Dataset:

8/8  1s 49ms/step - accuracy: 0.9363 - loss: 5.2468

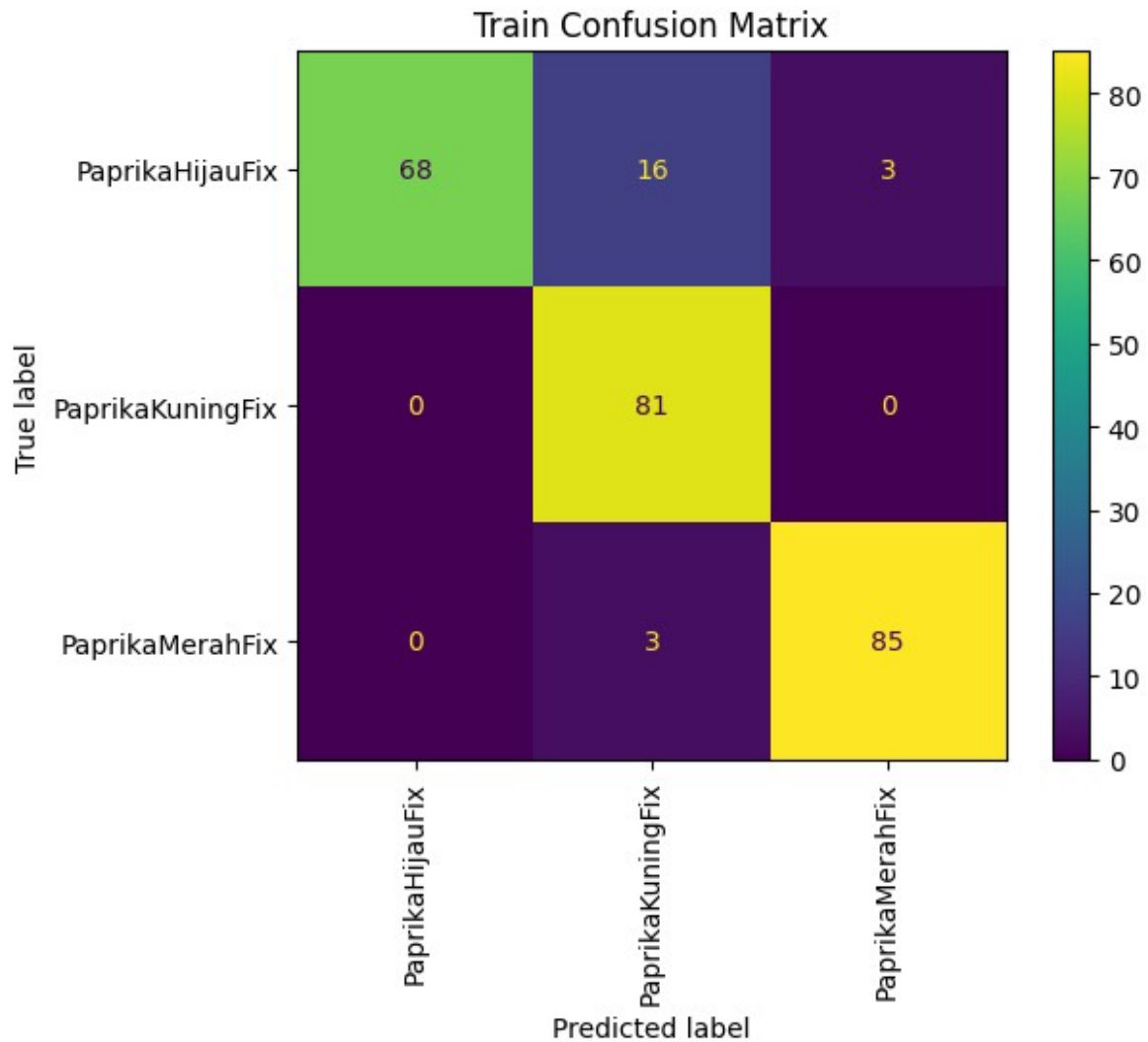
Train Accuracy: 0.93

Train Loss: 5.26

1/1  0s 247ms/step
 1/1  0s 93ms/step
 1/1  0s 94ms/step
 1/1  0s 98ms/step
 1/1  0s 101ms/step
 1/1  0s 97ms/step
 1/1  0s 100ms/step
 1/1  0s 100ms/step

Train Classification Report:

	precision	recall	f1-score	support
PaprikaHijauFix	1.00	0.78	0.88	87
PaprikaKuningFix	0.81	1.00	0.90	81
PaprikaMerahFix	0.97	0.97	0.97	88
accuracy			0.91	256
macro avg	0.93	0.92	0.91	256
weighted avg	0.93	0.91	0.91	256



Evaluating on Validation Dataset:

1/1 _____ 0s 108ms/step - accuracy: 0.9062 - loss: 5.3316

Validation Accuracy: 0.91

Validation Loss: 5.33

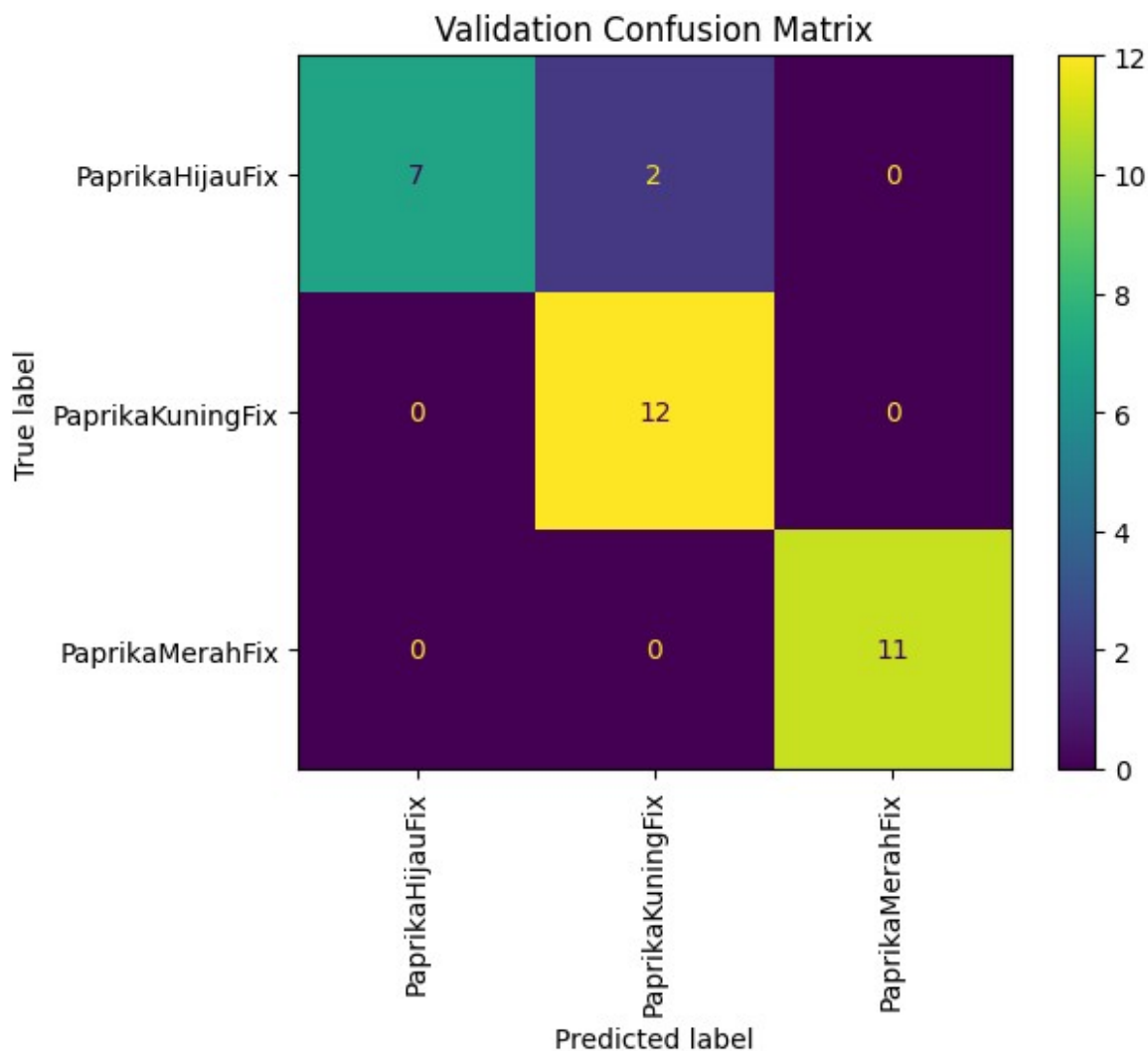
1/1 _____ 0s 97ms/step

Validation Classification Report:

	precision	recall	f1-score	support
PaprikaHijauFix	1.00	0.78	0.88	9
PaprikaKuningFix	0.86	1.00	0.92	12
PaprikaMerahFix	1.00	1.00	1.00	11
accuracy			0.94	32
macro avg	0.95	0.93	0.93	32

weighted avg	0.95	0.94	0.94	32
--------------	------	------	------	----

2024-12-19 20:20:31.291983: I
tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is
aborting with status: OUT_OF_RANGE: End of sequence



Evaluating on Test Dataset:

2/2

 0s 38ms/step - accuracy: 0.8948 - loss:
5.4322

Test Accuracy: 0.90

Test Loss: 5.39

1/1

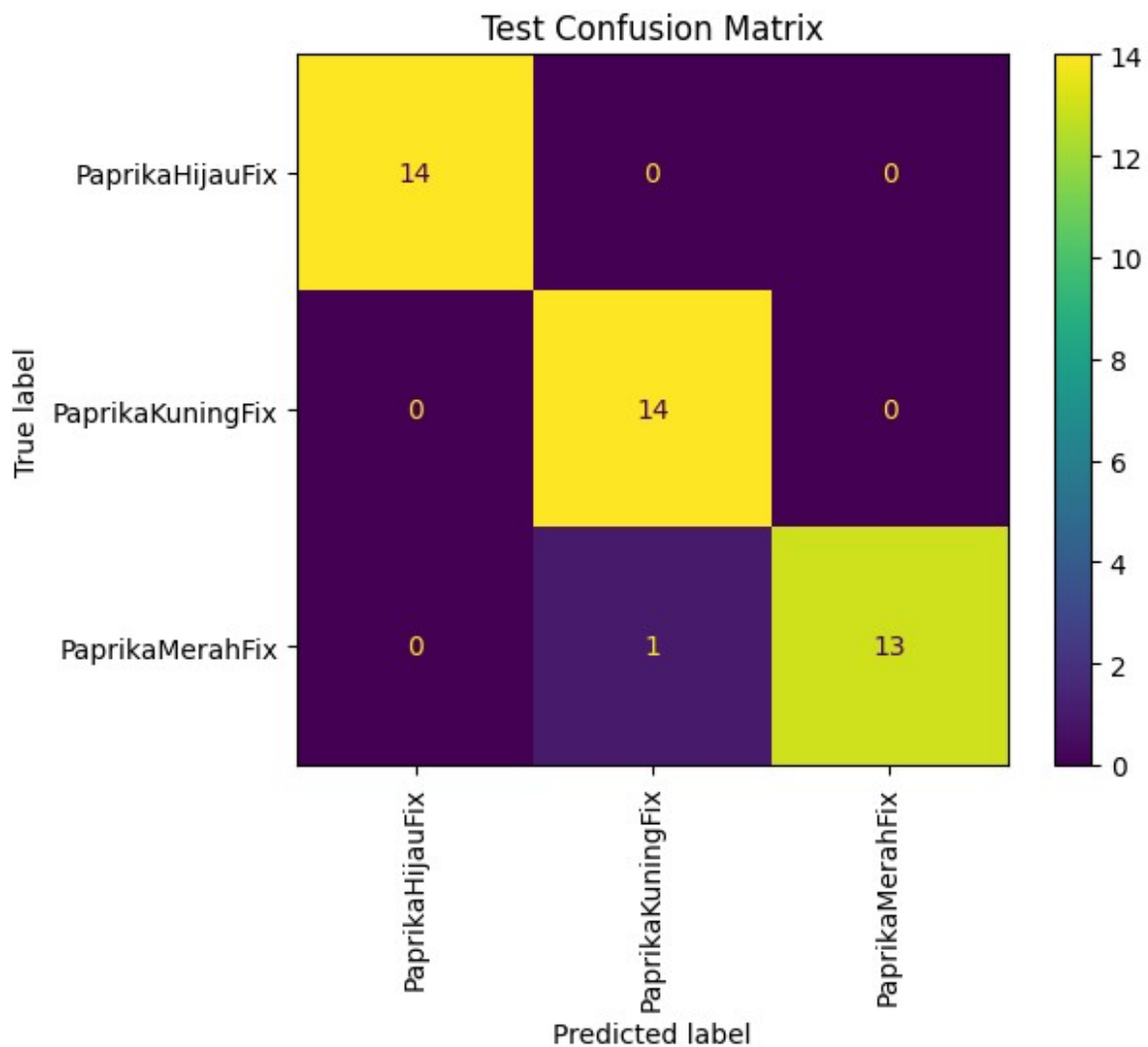
 0s 97ms/step

1/1

 0s 166ms/step

Test Classification Report:

	precision	recall	f1-score	support
PaprikaHijauFix	1.00	1.00	1.00	14
PaprikaKuningFix	0.93	1.00	0.97	14
PaprikaMerahFix	1.00	0.93	0.96	14
accuracy			0.98	42
macro avg	0.98	0.98	0.98	42
weighted avg	0.98	0.98	0.98	42



```
print("Made Ivan Jayasavitra Girinata / 220712063/ Seaborn/ Perbedaan  
Paprika Merah, Kuning Hijau Menggunakan AlexNet")
```

Made Ivan Jayasavitra Girinata / 220712063/ Seaborn/ Perbedaan Paprika
Merah, Kuning Hijau Menggunakan AlexNet

