

AlexNet - ADLI

```
import tensorflow as tf
import numpy as np
from matplotlib import pyplot as plt
data_dir = r"D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data"
data =
tf.keras.utils.image_dataset_from_directory(data_dir, seed=123, image_size=(180, 180), batch_size=16)
print(data.class_names)

class_names = data.class_names

Found 300 files belonging to 3 classes.
['Kacang Almond', 'Kacang Mete', 'Kacang Tanah']

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

Found 300 files belonging to 3 classes.

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

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

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

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

from PIL import Image
import os

def convert_images(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.lower().endswith(('.jpg', '.jpeg', '.png',
```

```

'.bmp')): # Pastikan format yang valid
    try:
        image_path = os.path.join(root, file)
        with Image.open(image_path) as img:
            new_image_path = image_path.replace(file,
file.split('.')[0] + '.jpg')
            img.convert('RGB').save(new_image_path,
'JPEG') # Mengonversi gambar ke JPEG
            print(f"Converted: {image_path} ->
{new_image_path}")
    except Exception as e:
        print(f"Error processing {image_path}: {e}")

```

```

base_dir = r'D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data'
convert_images(base_dir)

```

```

Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA1.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA1.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA10.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA10.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA100.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA100.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA11.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA11.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA12.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA12.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA13.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA13.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA14.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA14.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA15.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA15.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan

```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_90.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_91.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_91.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_92.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_92.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_93.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_93.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_94.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_94.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_95.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_95.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_96.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_96.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_97.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_97.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_98.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_98.jpg  
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan  
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_99.jpg  
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran  
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_99.jpg
```

```
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')
```

Kacang Tanah



Kacang Almond



Kacang Tanah



Kacang Mete



Kacang Mete



Kacang Tanah



Kacang Tanah



Kacang Tanah



Kacang Mete



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

(32, 180, 180, 3)

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

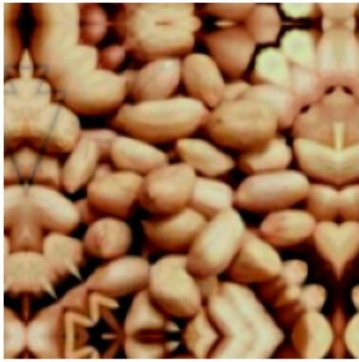
Tuner = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size =
Tuner)
```

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

data_augmentation = Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2),
    tf.keras.layers.RandomZoom(0.2),
    tf.keras.layers.RandomContrast(0.2),
    tf.keras.layers.RandomBrightness(0.2)
])

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

i = 0
plt.figure(figsize=(10,10))
for images, labels in train_ds.take(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')
```

```
from PIL import Image
import os

def convert_images(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.lower().endswith(('.jpg', '.jpeg', '.png',
            '.bmp')): # Pastikan format yang valid
                try:
                    image_path = os.path.join(root, file)
                    with Image.open(image_path) as img:
                        new_image_path = image_path.replace(file,
```

```

file.split('.')[0] + '.jpg')
        img.convert('RGB').save(new_image_path,
'JPEG') # Mengonversi gambar ke JPEG
        print(f"Converted: {image_path} ->
{new_image_path}")
    except Exception as e:
        print(f"Error processing {image_path}: {e}")

```

```

base_dir = r'D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data'
convert_images(base_dir)

```

```

Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA1.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA1.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA10.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA10.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA100.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA100.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA11.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA11.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA12.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA12.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA13.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA13.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA14.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA14.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA15.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA15.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA16.jpg -> D:\
Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\
UAS\train_data\Kacang Almond\KA16.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA17.jpg -> D:\

```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_91.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_92.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_92.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_93.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_93.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_94.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_94.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_95.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_95.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_96.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_96.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_97.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_97.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_98.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_98.jpg
Converted: D:\Matakuliah Semester 5\Pembelajaran Mesin dan
Pembelajaran Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_99.jpg
-> D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\train_data\Kacang Tanah\kacang_tanah_99.jpg

```
import tensorflow as tf
import keras
import keras._tf_keras.keras.backend as K
from keras._tf_keras.keras.models import Model
from keras._tf_keras.keras.layers import Input, Dense, Conv2D
from keras._tf_keras.keras.layers import Flatten, MaxPool2D, Dropout

# Membuat model AlexNet from scratch
def alexnet(input_shape, n_classes):

    input = Input(input_shape)

    # Layer 1
    x = Conv2D(96, (11, 11), strides=4, activation='relu')(input)
    x = MaxPool2D(pool_size=(3, 3), strides=2)(x)
```

```

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

# Layer 3
x = Conv2D(384, (3, 3), padding='same', activation='relu')(x)

# Layer 4
x = Conv2D(384, (3, 3), padding='same', activation='relu')(x)

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

# Membuat model
model = Model(inputs=input, outputs=output)
return model

# Pastikan input shape dan jumlah kelas sesuai
input_shape = (180, 180, 3) # Resolusi gambar Anda
n_classes = 2 # Jumlah kelas (misalnya: kacang tanah dan kacang mete)

# Clear cache Keras menggunakan clear session
K.clear_session()

# Membuat model AlexNet
model = alexnet(input_shape, n_classes)

# Menampilkan summary dari model
model.summary()

Model: "functional"

```

Layer (type)	Output Shape
Param #	
input_layer (InputLayer)	(None, 180, 180, 3)

0				
		conv2d (Conv2D)	(None, 43, 43, 96)	
34,944				
		max_pooling2d (MaxPooling2D)	(None, 21, 21, 96)	
0				
		conv2d_1 (Conv2D)	(None, 21, 21, 256)	
614,656				
		max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 256)	
0				
		conv2d_2 (Conv2D)	(None, 10, 10, 384)	
885,120				
		conv2d_3 (Conv2D)	(None, 10, 10, 384)	
1,327,488				
		conv2d_4 (Conv2D)	(None, 10, 10, 256)	
884,992				
		max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 256)	
0				
		flatten (Flatten)	(None, 4096)	
0				
		dense (Dense)	(None, 4096)	
16,781,312				
		dropout (Dropout)	(None, 4096)	
0				
		dense_1 (Dense)	(None, 4096)	
16,781,312				

dropout_1 (Dropout)	(None, 4096)	
0		
dense_2 (Dense)	(None, 2)	
8,194		

Total params: 37,318,018 (142.36 MB)

Trainable params: 37,318,018 (142.36 MB)

Non-trainable params: 0 (0.00 B)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
```

Model AlexNet dengan Sequential

```
model = Sequential([
    # Layer 1: Convolution + MaxPooling
    Conv2D(96, (11, 11), strides=4, activation='relu',
input_shape=(180, 180, 3)),
    MaxPool2D(pool_size=(3, 3), strides=2),

    # Layer 2: Convolution + MaxPooling
    Conv2D(256, (5, 5), padding='same', activation='relu'),
    MaxPool2D(pool_size=(3, 3), strides=2),

    # Layer 3: Convolution
    Conv2D(384, (3, 3), padding='same', activation='relu'),

    # Layer 4: Convolution
    Conv2D(384, (3, 3), padding='same', activation='relu'),

    # Layer 5: Convolution + MaxPooling
    Conv2D(256, (3, 3), padding='same', activation='relu'),
    MaxPool2D(pool_size=(3, 3), strides=2),

    # Flatten before fully connected layers
    Flatten(),

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

```

Dense(4096, activation='relu'),
Dropout(0.5),

# Output layer
Dense(3, activation='softmax') # Jumlah kelas: 3
])

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

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

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

Epoch 1/30
9/9 _____ 26s 2s/step - accuracy: 0.3965 - loss: 7.3794
- val_accuracy: 0.4167 - val_loss: 1.2820
Epoch 2/30
9/9 _____ 19s 2s/step - accuracy: 0.3488 - loss: 1.7135
- val_accuracy: 0.6667 - val_loss: 0.8795
Epoch 3/30
9/9 _____ 19s 2s/step - accuracy: 0.4479 - loss: 1.1557
- val_accuracy: 0.7500 - val_loss: 0.7952
Epoch 4/30
9/9 _____ 19s 2s/step - accuracy: 0.6836 - loss: 0.7921
- val_accuracy: 0.7500 - val_loss: 0.9462
Epoch 5/30
9/9 _____ 19s 2s/step - accuracy: 0.6488 - loss: 0.7946
- val_accuracy: 0.5833 - val_loss: 0.8338
Epoch 6/30
9/9 _____ 20s 2s/step - accuracy: 0.7385 - loss: 0.6202
- val_accuracy: 0.5833 - val_loss: 0.6658
Epoch 7/30
9/9 _____ 20s 2s/step - accuracy: 0.7773 - loss: 0.5918

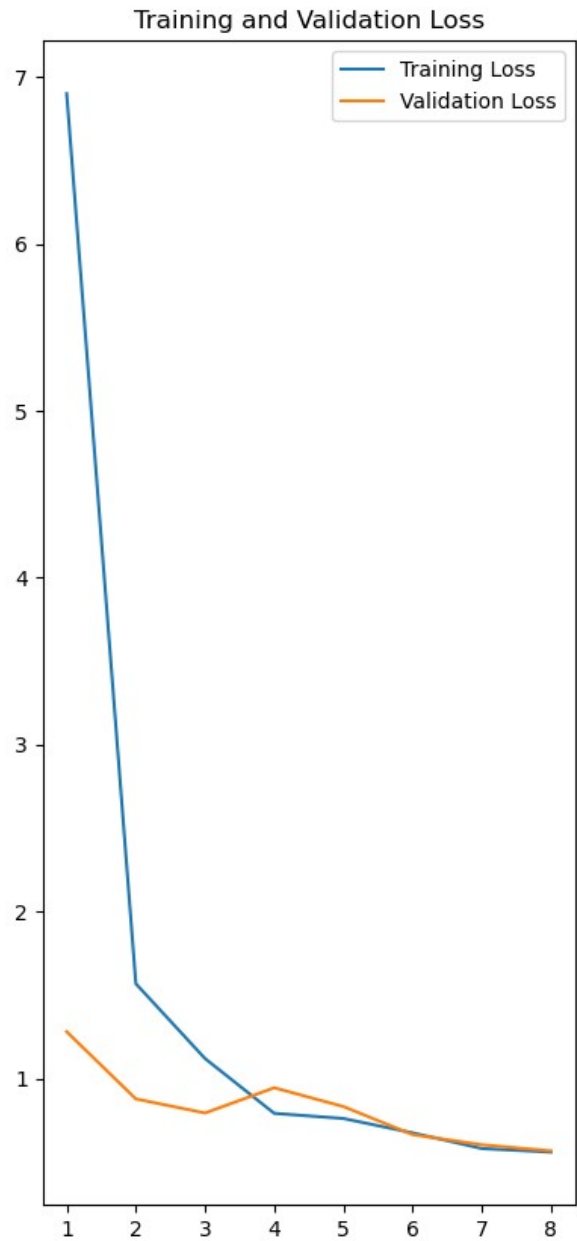
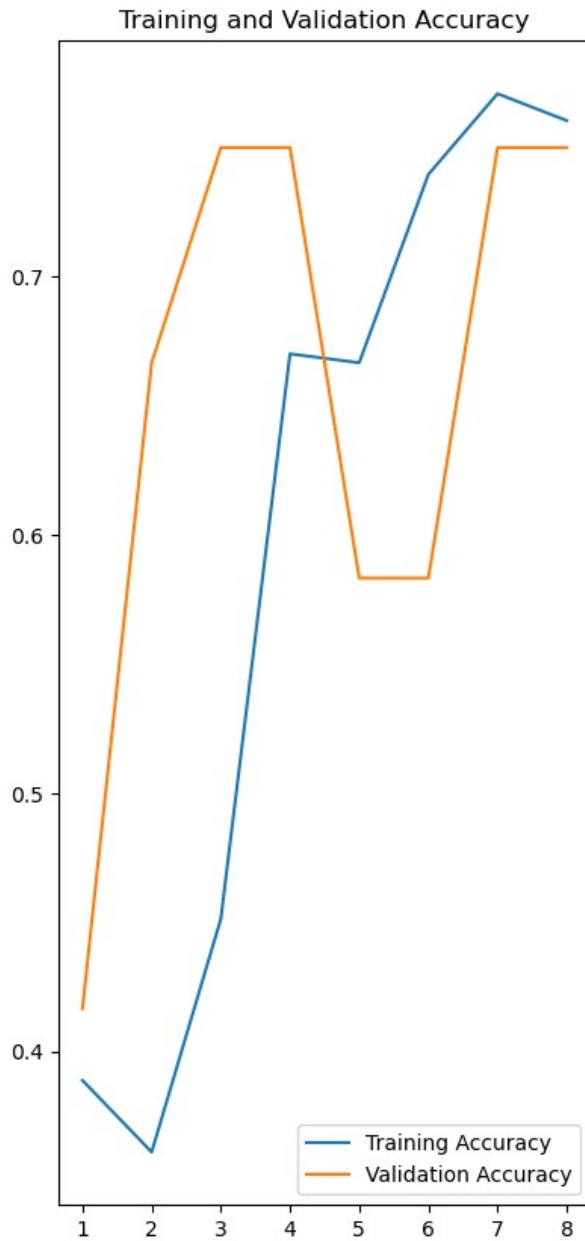
```

```
- val_accuracy: 0.7500 - val_loss: 0.6050
Epoch 8/30
9/9 _____ 19s 2s/step - accuracy: 0.7451 - loss: 0.5821
- val_accuracy: 0.7500 - val_loss: 0.5683
```

#buat plot dengan menggunakan history supaya jumlahnya sesuai epoch yang dilakukan

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

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



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

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

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

```

from PIL import Image

# Load the trained model
model = load_model(r'BestModel_AlexNet_Numpy.h5') # Ganti dengan path model Anda
class_names = ['Kacang Almond', 'Kacang Mete', 'Kacang Tanah']

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

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

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

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

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

# Contoh penggunaan fungsi
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
result = classify_images(r'D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\UAS\train_data\Kacang Almond\KA3.jpg',
save_path='kacang almond.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.

WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x00000291DEF0AAC0> 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 that can 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 5 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x00000291DEF0AAC0> 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 that can 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.

1/1 ————— 0s 237ms/step

Prediksi: Kacang Almond

Confidence: 46.19%

Prediksi: Kacang Almond dengan confidence 46.19%. Gambar asli disimpan di kacang almond.jpg.

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

# Muat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'D:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran
Mendalam\UAS\test_data',
    labels='inferred',
    label_mode='categorical', # Menghasilkan label dalam bentuk one-
hot encoding
    batch_size=32,
    image_size=(180, 180)
)

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

# Ekstrak label sebenarnya dari test_data dan konversi ke bentuk
```

```

indeks_kelas
true_labels = []
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy()) # Konversi
one-hot ke indeks kelas
true_labels = tf.convert_to_tensor(true_labels)

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

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

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

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

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

# Menampilkan hasil
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
print("Confusion Matrix:\n", conf_mat.numpy())
print("Akurasi:", accuracy.numpy())
print("Presisi:", precision.numpy())
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())

```

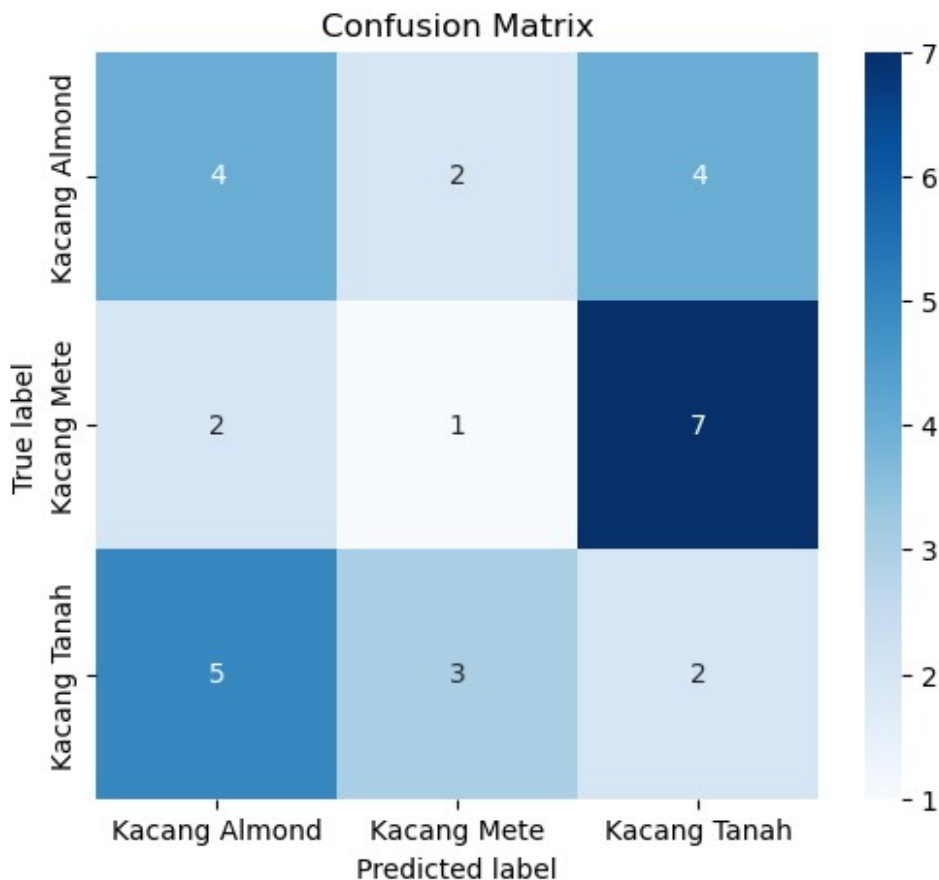
Found 30 files belonging to 3 classes.

WARNING:tensorflow:6 out of the last 6 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x00000291DEF0AAC0> 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 that can 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:6 out of the last 6 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x00000291DEF0AAC0> 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 that can 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.

1/1 ————— 2s 2s/step



Confusion Matrix:

[[4 2 4]

[2 1 7]

[5 3 2]]

Akurasi: 0.23333333333333334

Presisi: [0.36363636 0.16666667 0.15384615]

Recall: [0.4 0.1 0.2]

F1 Score: [0.38095238 0.125 0.17391304]

```

import tensorflow as tf
import numpy as np
from matplotlib import pyplot as plt
data_dir = r"C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data"
data =
tf.keras.utils.image_dataset_from_directory(data_dir, seed=123, image_size=(180,180), batch_size=16)
print(data.class_names)

```

```
class_names = data.class_names
```

```

Found 300 files belonging to 3 classes.
['Kacang Almond', 'Kacang Mete', 'Kacang Tanah']

```

```

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

```

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

```

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

```

```

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

```

```

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

```

```

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

```

```

from PIL import Image
import os

```

```

def convert_images(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.lower().endswith(('.jpg', '.jpeg', '.png',
            '.bmp')): # Pastikan format yang valid
                try:
                    image_path = os.path.join(root, file)

```

```

        with Image.open(image_path) as img:
            new_image_path = image_path.replace(file,
file.split('.')[0] + '.jpg')
            img.convert('RGB').save(new_image_path,
'JPEG') # Mengonversi gambar ke JPEG
            print(f"Converted: {image_path} ->
{new_image_path}")
        except Exception as e:
            print(f"Error processing {image_path}: {e}")

base_dir = r'C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data'
convert_images(base_dir)

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA1.jpg -> C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\
train_data\Kacang Almond\KA1.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA10.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA10.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA100.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA100.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA11.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA11.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA12.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA12.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA13.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA13.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA14.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA14.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA15.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA15.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA16.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA16.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA17.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA17.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA18.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA18.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA19.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA19.jpg

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\

```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
Kacang Tanah\kacang_tanah_90.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_90.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_91.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_91.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_92.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_92.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_93.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_93.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_94.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_94.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_95.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_95.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_96.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_96.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_97.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_97.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_98.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_98.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_99.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_99.jpg
```

```
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')
```


Kacang Tanah



Kacang Almond



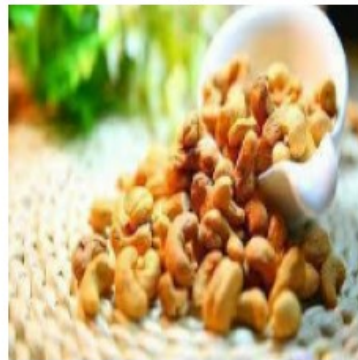
Kacang Tanah



Kacang Mete



Kacang Mete



Kacang Tanah



Kacang Tanah



Kacang Tanah



Kacang Mete



```
for images, labels in train_ds.take(1):  
    images_array = np.array(images)  
    print(images_array.shape)  
  
(32, 180, 180, 3)  
  
from tensorflow.keras import layers  
from tensorflow.keras.models import Sequential, load_model  
  
Tuner = tf.data.AUTOTUNE  
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size =  
Tuner)
```

```

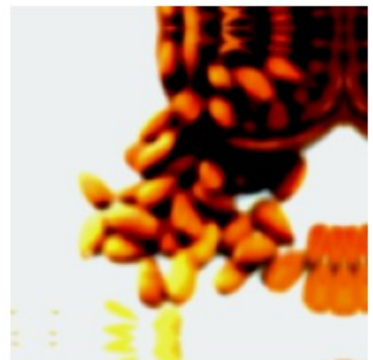
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = Tuner)

data_augmentation = Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2),
    tf.keras.layers.RandomZoom(0.2),
    tf.keras.layers.RandomContrast(0.2),
    tf.keras.layers.RandomBrightness(0.2)
])

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

i = 0
plt.figure(figsize=(10,10))
for images, labels in train_ds.take(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')

```



```
from PIL import Image
import os

def convert_images(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.lower().endswith(('.jpg', '.jpeg', '.png',
            '.bmp')): # Pastikan format yang valid
                try:
                    image_path = os.path.join(root, file)
                    with Image.open(image_path) as img:
                        new_image_path = image_path.replace(file,
```



```

file.split('.')[0] + '.jpg')
        img.convert('RGB').save(new_image_path,
'JPEG') # Mengonversi gambar ke JPEG
        print(f"Converted: {image_path} ->
{new_image_path}")
    except Exception as e:
        print(f"Error processing {image_path}: {e}")

base_dir = r'C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data'
convert_images(base_dir)

```

```

Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA1.jpg -> C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\
train_data\Kacang Almond\KA1.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA10.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA10.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA100.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA100.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA11.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA11.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA12.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA12.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA13.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA13.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA14.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA14.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA15.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA15.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA16.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA16.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA17.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA17.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA18.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA18.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA19.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Almond\KA19.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Almond\KA2.jpg -> C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\
train_data\Kacang Almond\KA2.jpg

```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_91.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_91.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_92.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_92.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_93.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_93.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_94.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_94.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_95.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_95.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_96.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_96.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_97.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_97.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_98.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_98.jpg
Converted: C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\train_data\
Kacang Tanah\kacang_tanah_99.jpg -> C:\Users\PC\Downloads\
Googlenet_A_Numpy_RAFI\train_data\Kacang Tanah\kacang_tanah_99.jpg
```

```
import tensorflow as tf
from keras._tf_keras.keras.models import Model
from keras._tf_keras.keras.layers import Input, Conv2D, MaxPool2D,
Flatten, Dense, Dropout
import keras._tf_keras.keras.backend as K

# Membuat model VGG-16 dari awal
def vgg16(input_shape, n_classes):
    input = Input(shape=input_shape)

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

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

    # Block 3
    x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)
```

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

```
# Block 4
```

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

```
# Block 5
```

```
x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
x = MaxPool2D((2, 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)
```

```
# Membuat model
```

```
model = Model(inputs=input, outputs=output)
return model
```

```
# Pastikan input shape dan jumlah kelas sesuai
```

```
input_shape = (180, 180, 3) # Resolusi gambar
```

```
n_classes = 3 # Contoh: kacang tanah, kacang mete, kacang almond
```

```
# Clear cache Keras menggunakan clear session
```

```
K.clear_session()
```

```
# Membuat model VGG-16
```

```
model = vgg16(input_shape, n_classes)
```

```
# Menampilkan summary dari model
```

```
model.summary()
```

```
WARNING:tensorflow:From c:\Users\PC\anaconda3\Lib\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) Param #	Output Shape	
input_layer (InputLayer) 0	(None, 180, 180, 3)	
conv2d (Conv2D) 1,792	(None, 180, 180, 64)	
conv2d_1 (Conv2D) 36,928	(None, 180, 180, 64)	
max_pooling2d (MaxPooling2D) 0	(None, 90, 90, 64)	
conv2d_2 (Conv2D) 73,856	(None, 90, 90, 128)	
conv2d_3 (Conv2D) 147,584	(None, 90, 90, 128)	
max_pooling2d_1 (MaxPooling2D) 0	(None, 45, 45, 128)	
conv2d_4 (Conv2D) 295,168	(None, 45, 45, 256)	
conv2d_5 (Conv2D) 590,080	(None, 45, 45, 256)	
conv2d_6 (Conv2D) 590,080	(None, 45, 45, 256)	
max_pooling2d_2 (MaxPooling2D) 0	(None, 22, 22, 256)	

conv2d_7 (Conv2D)	(None, 22, 22, 512)	
1,180,160		
conv2d_8 (Conv2D)	(None, 22, 22, 512)	
2,359,808		
conv2d_9 (Conv2D)	(None, 22, 22, 512)	
2,359,808		
max_pooling2d_3 (MaxPooling2D)	(None, 11, 11, 512)	
0		
conv2d_10 (Conv2D)	(None, 11, 11, 512)	
2,359,808		
conv2d_11 (Conv2D)	(None, 11, 11, 512)	
2,359,808		
conv2d_12 (Conv2D)	(None, 11, 11, 512)	
2,359,808		
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 512)	
0		
flatten (Flatten)	(None, 12800)	
0		
dense (Dense)	(None, 4096)	
52,432,896		
dropout (Dropout)	(None, 4096)	
0		
dense_1 (Dense)	(None, 4096)	
16,781,312		
dropout_1 (Dropout)	(None, 4096)	
0		

dense_2 (Dense)	(None, 3)	
12,291		

Total params: 83,941,187 (320.21 MB)

Trainable params: 83,941,187 (320.21 MB)

Non-trainable params: 0 (0.00 B)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense,
Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
```

Model VGG-16 dengan Sequential

```
model = Sequential([
    # Block 1
    Conv2D(64, (3, 3), padding='same', activation='relu',
input_shape=(180, 180, 3)),
    Conv2D(64, (3, 3), padding='same', activation='relu'),
    MaxPool2D(pool_size=(2, 2), strides=2),

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

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

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

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

Fully connected layers

```

        Flatten(),
        Dense(4096, activation='relu'),
        Dropout(0.5),
        Dense(4096, activation='relu'),
        Dropout(0.5),

        # Output layer
        Dense(3, activation='softmax') # Jumlah kelas: 3
    ])

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

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

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

c:\Users\PC\anaconda3\Lib\site-packages\keras\src\layers\
convolutional\base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/30
9/9 _____ 59s 6s/step - accuracy: 0.3695 - loss: 1.2266
- val_accuracy: 0.1667 - val_loss: 1.1342
Epoch 2/30
9/9 _____ 53s 6s/step - accuracy: 0.3442 - loss: 1.0957
- val_accuracy: 0.3333 - val_loss: 1.0365
Epoch 3/30
9/9 _____ 55s 6s/step - accuracy: 0.3968 - loss: 1.0466
- val_accuracy: 0.4167 - val_loss: 1.0272

```



```

Epoch 4/30
9/9 _____ 54s 6s/step - accuracy: 0.5657 - loss: 0.8610
- val_accuracy: 0.5000 - val_loss: 1.3523
Epoch 5/30
9/9 _____ 57s 6s/step - accuracy: 0.6060 - loss: 0.7995
- val_accuracy: 0.5000 - val_loss: 0.9711
Epoch 6/30
9/9 _____ 55s 6s/step - accuracy: 0.6237 - loss: 0.7718
- val_accuracy: 0.5833 - val_loss: 1.0156
Epoch 7/30
9/9 _____ 53s 6s/step - accuracy: 0.6508 - loss: 0.7727
- val_accuracy: 0.4167 - val_loss: 0.8240
Epoch 8/30
9/9 _____ 56s 6s/step - accuracy: 0.7047 - loss: 0.7282
- val_accuracy: 0.6667 - val_loss: 0.7695
Epoch 9/30
9/9 _____ 57s 6s/step - accuracy: 0.7350 - loss: 0.7071
- val_accuracy: 0.6667 - val_loss: 0.6958
Epoch 10/30
9/9 _____ 60s 7s/step - accuracy: 0.6265 - loss: 0.7974
- val_accuracy: 0.7500 - val_loss: 0.8145
Epoch 11/30
9/9 _____ 55s 6s/step - accuracy: 0.6864 - loss: 0.7406
- val_accuracy: 0.7500 - val_loss: 0.6792
Epoch 12/30
9/9 _____ 57s 6s/step - accuracy: 0.7618 - loss: 0.5754
- val_accuracy: 0.6667 - val_loss: 0.6227
Epoch 13/30
9/9 _____ 55s 6s/step - accuracy: 0.8153 - loss: 0.5189
- val_accuracy: 0.9167 - val_loss: 0.4425
Epoch 14/30
9/9 _____ 60s 7s/step - accuracy: 0.7626 - loss: 0.6096
- val_accuracy: 0.5833 - val_loss: 0.6788
Epoch 15/30
9/9 _____ 54s 6s/step - accuracy: 0.7515 - loss: 0.6080
- val_accuracy: 0.5833 - val_loss: 0.7454
Epoch 16/30
9/9 _____ 61s 7s/step - accuracy: 0.7676 - loss: 0.5352
- val_accuracy: 0.9167 - val_loss: 0.3456
Epoch 17/30
9/9 _____ 55s 6s/step - accuracy: 0.8320 - loss: 0.4581
- val_accuracy: 0.9167 - val_loss: 0.3379
Epoch 18/30
9/9 _____ 53s 6s/step - accuracy: 0.8340 - loss: 0.4463
- val_accuracy: 0.9167 - val_loss: 0.3387

```

#buat plot dengan menggunakan history supaya jumlahnya sesuai epoch yang dilakukan

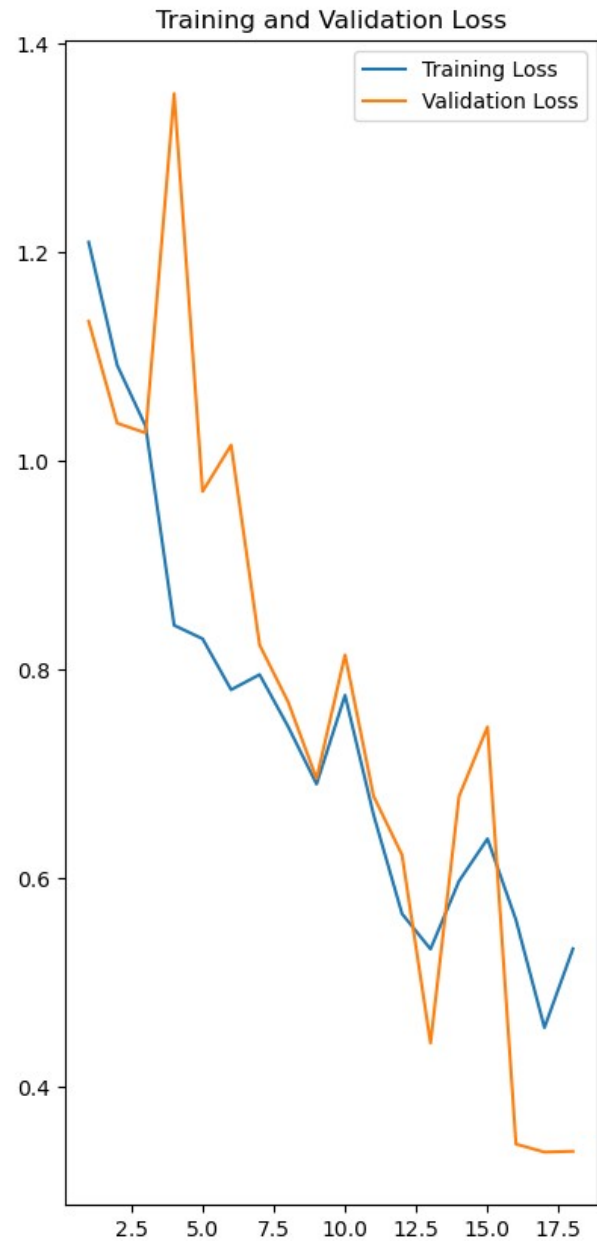
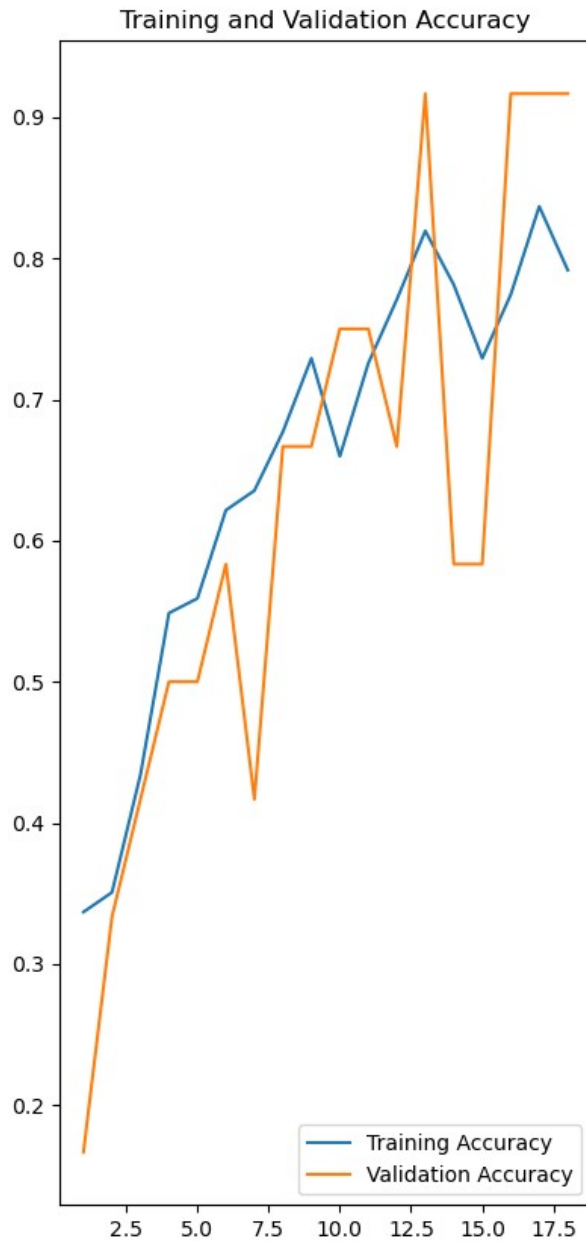
```

ephocs_range = range(1, len(history.history['loss']) + 1)
plt.figure(figsize=(10, 10))

```

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

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



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

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

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

```

from PIL import Image

# Load the trained model
model = load_model(r'BestModel_VGG-16_Numpy.h5') # Ganti dengan path model Anda
class_names = ['Kacang Almond', 'Kacang Mete', 'Kacang Tanah']

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

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

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

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

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

# Contoh penggunaan fungsi
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
result = classify_images(r'C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\test_data\Validation Kacang Almond\KA3.jpg',
save_path='KA3.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.

Terjadi kesalahan: [Errno 2] No such file or directory: 'C:\\Users\\PC\\Downloads\\Googlenet_A_Numpy_RAFI\\test_data\\Validation Kacang Almond\\KA3.jpg'

```

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

# Muat data test yang sebenarnya
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\PC\Downloads\Googlenet_A_Numpy_RAFI\test_data',
    labels='inferred',
    label_mode='categorical', # Menghasilkan label dalam bentuk one-
hot encoding
    batch_size=32,
    image_size=(180, 180)
)

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

# Ekstrak label sebenarnya dari test_data dan konversi ke bentuk
indeks kelas
true_labels = []
for _, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy()) # Konversi
one-hot ke indeks kelas
true_labels = tf.convert_to_tensor(true_labels)

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

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

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

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

# Visualisasi Confusion Matrix
plt.figure(figsize=(6, 5))
sns.heatmap(conf_mat.numpy(), annot=True, fmt='d', cmap='Blues',
            xticklabels=["Kacang Almond", "Kacang Mete", "Kacang
Tanah"], yticklabels=["Kacang Almond", "Kacang Mete", "Kacang Tanah"])
plt.title('Confusion Matrix')

```

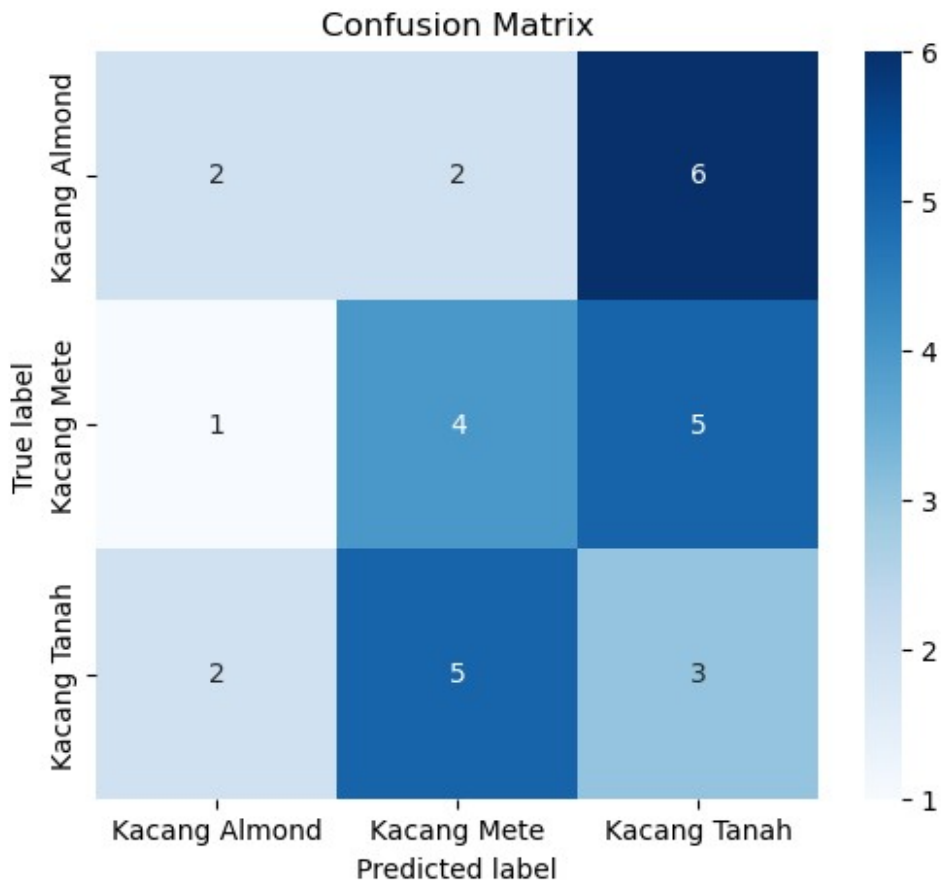
```

plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Menampilkan hasil
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
print("Confusion Matrix:\n", conf_mat.numpy())
print("Akurasi:", accuracy.numpy())
print("Presisi:", precision.numpy())
print("Recall:", recall.numpy())
print("F1 Score:", f1_score.numpy())

```

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



```

Confusion Matrix:
[[2 2 6]
 [1 4 5]
 [2 5 3]]
Akurasi: 0.3
Presisi: [0.4      0.36363636 0.21428571]

```

Recall: [0.2 0.4 0.3]

F1 Score: [0.26666667 0.38095238 0.25]

```

#Import library
import os
import numpy as np

#Import library tensorflow dan modul keras yang diperlukan
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import load_img,
ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
Dropout, Flatten, Rescaling, GlobalAveragePooling2D, RandomFlip,
RandomRotation, RandomZoom

#Penjelasan
# layers digunakan untuk menambahkan lapisan ke dalam model
# load_img digunakan untuk memuat gambar
# ImageDataGenerator digunakan untuk melakukan augmentasi pada gambar
# Sequential digunakan untuk membuat model secara berurutan
# Conv2D digunakan untuk membuat lapisan konvolusi
# MaxPooling2D digunakan untuk melakukan pooling pada lapisan
konvolusi
# Dense digunakan untuk membuat lapisan fully connected
# Dropout digunakan untuk menghindari overfitting
# Flatten digunakan untuk membuat lapisan menjadi flat (rata) menjadi
vektor 1 dimensi

count = 0 #digunakan untuk menghitung jumlah gambar
dirs = os.listdir(r'D:\Nicolaus Kevin PH\SEMS 5\ML\UAS\train_data')
for dir in dirs:
    files = list(os.listdir(r'D:\Nicolaus Kevin PH\SEMS 5\ML\UAS\
train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')

Kacang Almond Folder has 100 Images
Kacang Mete Folder has 100 Images
Kacang Tanah Folder has 100 Images
Images Folder has 300 Images

# Parameter
base_dir = r'D:\Nicolaus Kevin PH\SEMS 5\ML\UAS\train_data' #direktori
folder dataset
img_size = 180 #mengubah ukuran gambar menjadi 180
batch = 10 #jumlah sample (gambar) yang akan diproses pada satu kali
iterasi
validation_split = 0.1 #data pelatihan yang akan digunakan sebagai
data validasi

```



```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir, #path direktori, subfolder dianggap sebagai label
    seed=96, #untuk memastikan proses pemisahan data selalu konsisten
    (random_state)
    image_size=(img_size, img_size), #ukuran gambar diubah (resize)
    menjadi 180x180 pixel
    batch_size=batch, #jumlah gambar yang akan dikelompokkan
)
```

Found 300 files belonging to 3 classes.

```
#mendapatkan nama kelas dari dataset
class_names = dataset.class_names #dataset.class_names akan mengambil
daftar nama kelas berdasarkan subfolder di dalam direktori
print("Class Names:", class_names)
```

```
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
menghitung jumlah gambar untuk train
```

```
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)
```

```
Class Names: ['Kacang Almond', 'Kacang Mete', 'Kacang Tanah']
Total Images: 30
Train Images: 27
Validation Images: 3
```

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

```
import matplotlib.pyplot as plt
```

```
i = 0
plt.figure(figsize=(10,10)) #membuat figure dengan ukuran 10x10 inchi
untuk menampilkan gambar
```

```
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
for images, labels in train_ds.take(1): #mengambil 1 batch pertama
    dari train_ds
```

```
        for i in range(9):
            plt.subplot(3,3, i+1) #menyiapkan subplot dengan grid 3x3 dan
            menempatkan gambar pada posisi i+1
            plt.imshow(images[i].numpy().astype('uint8')) #menampilkan
            gambar dan mengonversi ke tipe uint8
            plt.title(class_names[labels[i]]) #menampilkan judul gambar
            sesuai dengan nama kelas
```

```
plt.axis('off') #menonaktifkan sumbu pada gambar agar tidak terlihat
```

Kacang Mete



Kacang Mete



Kacang Mete



Kacang Mete



Kacang Tanah



Kacang Mete



Kacang Almond



Kacang Almond



Kacang Almond



```
import numpy as np

# Tampilkan gambar dengan shape (32, 180, 180, 3)
for images, labels in train_ds.take(1):
    images_array = np.array(images)
    print(images_array.shape) # Output: (32, 180, 180, 3)
    #32: Jumlah gambar dalam batch.
    #180: Lebar gambar dalam piksel
```

#180: Tinggi gambar dalam piksel
#3: Jumlah channel gambar (RGB)

```
(10, 180, 180, 3)
```

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

```
data_augmentation = Sequential([  
    layers.RandomFlip("horizontal", input_shape =  
(img_size,img_size,3)), #membalik gambar secara horizontal  
    layers.RandomRotation(0.1), #merotasi gambar secara acak dalam  
kisaran 0°-36° (0.1 * 360)  
    layers.RandomZoom(0.1) #melakukan zoom in/zoom out secara acak  
dengan rentang 10%  
])
```

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

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

*#sama seperti sebelumnya, code ini digunakan untuk menampilkan gambar
dari data_augmentation*

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

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




```
#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, # Exclude fully connected layers
```

```

    input_shape=(img_size, img_size, 3) # Input shape
)

#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

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

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
)

C:\Users\M S I\AppData\Local\Temp\ipykernel_21980\469119848.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(

model.summary()

Model: "sequential_84"

```

Layer (type)	Output Shape
Param #	

0	sequential_83 (Sequential)	(None, 180, 180, 3)	
0	rescaling_39 (Rescaling)	(None, 180, 180, 3)	
3,228,864	mobilenet_1.00_224 (Functional)	(None, 5, 5, 1024)	
0	global_average_pooling2d_36 (GlobalAveragePooling2D)	(None, 1024)	
131,200	dense_75 (Dense)	(None, 128)	
0	dropout_39 (Dropout)	(None, 128)	
387	dense_76 (Dense)	(None, 3)	

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

Trainable params: 3,069,443 (11.71 MB)

Non-trainable params: 291,008 (1.11 MB)

#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

27/27 _____ 14s 176ms/step - accuracy: 0.5946 - loss:
0.9728 - val_accuracy: 0.2667 - val_loss: 2.8378

Epoch 2/30

27/27 _____ 4s 145ms/step - accuracy: 0.8898 - loss:
0.3040 - val_accuracy: 0.3000 - val_loss: 2.5967

Epoch 3/30

27/27 _____ 4s 145ms/step - accuracy: 0.9732 - loss:
0.1414 - val_accuracy: 0.6333 - val_loss: 1.1486

Epoch 4/30

27/27 _____ 4s 145ms/step - accuracy: 0.9837 - loss:
0.0875 - val_accuracy: 0.8333 - val_loss: 0.4280

Epoch 5/30

27/27 _____ 4s 145ms/step - accuracy: 0.9628 - loss:
0.0964 - val_accuracy: 0.8667 - val_loss: 0.2373

Epoch 6/30

27/27 _____ 4s 152ms/step - accuracy: 0.9863 - loss:
0.0373 - val_accuracy: 0.9333 - val_loss: 0.1255

Epoch 7/30

27/27 _____ 4s 153ms/step - accuracy: 0.9937 - loss:
0.0385 - val_accuracy: 0.9667 - val_loss: 0.0735

Epoch 8/30

27/27 _____ 5s 169ms/step - accuracy: 0.9934 - loss:
0.0239 - val_accuracy: 0.9667 - val_loss: 0.0775

Epoch 9/30

27/27 _____ 4s 158ms/step - accuracy: 0.9968 - loss:
0.0242 - val_accuracy: 0.9667 - val_loss: 0.0659

Epoch 10/30

27/27 _____ 4s 158ms/step - accuracy: 0.9997 - loss:
0.0148 - val_accuracy: 0.9667 - val_loss: 0.0340

*#membuat range untuk epoch berdasarkan panjang data loss dari
pelatihan*

`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)*

#grafik pertama (Training and Validation Accuracy)

`plt.subplot(1, 2, 1)` *#membuat subplot pertama dalam layout 1 baris dan
2 kolom*

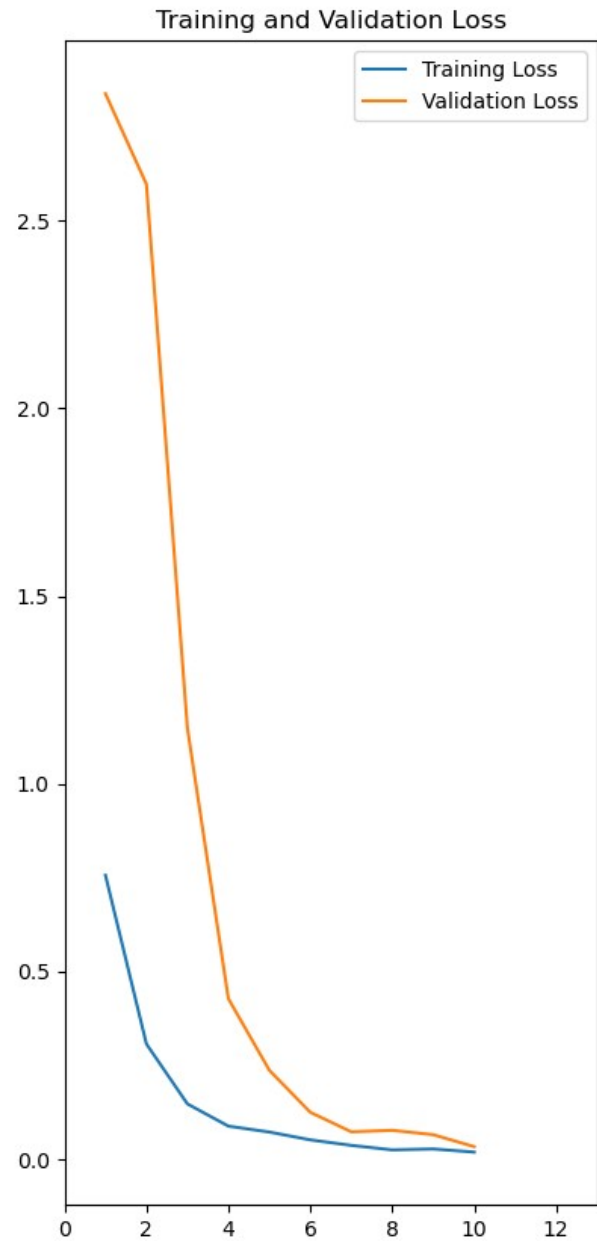
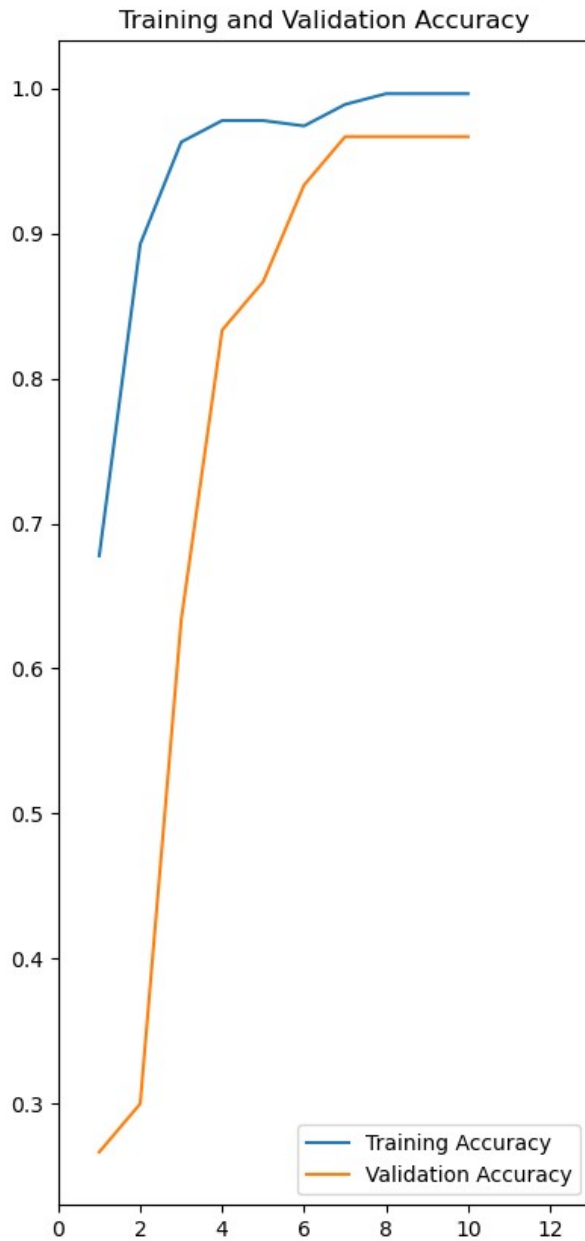
`plt.plot(ephocs_range, history.history['accuracy'], label='Training
Accuracy')` *#plot akurasi pelatihan*

`plt.plot(ephocs_range, history.history['val_accuracy'],
label='Validation Accuracy')` *#plot akurasi validasi*

`plt.legend(loc='lower right')` *#membuat legenda (informasi elemen*

```
visual) di sudut kanan bawah
plt.xlim(0, 13) #mengatur batas nilai pada sumbu x dari epoch 1 sampai
13
plt.title('Training and Validation Accuracy') #memberi judul grafik

#grafik kedua (Training and Validation Loss)
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()
```

```
#menyimpan model yang telah dilatih
model.save('BestModel_MobileNet_Numpy.h5')
```

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

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

```

from tensorflow.keras.models import load_model
from PIL import Image

#memuat model yang sudah dilatih
model = load_model(r'D:\Nicolaus Kevin PH\SEMS 5\ML\UAS\
BestModel_MobileNet_Numpy.h5') # Ganti dengan path model Anda
class_names = ['Kacang Mete', 'Kacang Almond', 'Kacang Tanah'] #kelas
yang ada pada model

#fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        #memuat dan mempersiapkan gambar untuk prediksi
        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
###Terdapat code yang hilang disini! lihat modul untuk menemukanya
result = classify_images(r'D:\Nicolaus Kevin PH\SEMS 5\ML\UAS\
test_data\Validation Kacang Mete\image 3.jpg', save_path='METE.jpg')
print(result)
```

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

1/1 _____ 1s 654ms/step

Prediksi: Kacang Almond

Confidence: 57.02%

Prediksi: Kacang Almond dengan confidence 57.02%. Gambar asli disimpan di METE.jpg.

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

#memuat model yang telah dilatih sebelumnya

```
mobileNet_model = load_model(r'D:\Nicolaus Kevin PH\SEMS 5\ML\Pert 13\
Tugas6_A_11796\model_mobilenet.h5')#gunakan path masing masing ya
```

#memuat data test yang sebenarnya

```
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'test_data', #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

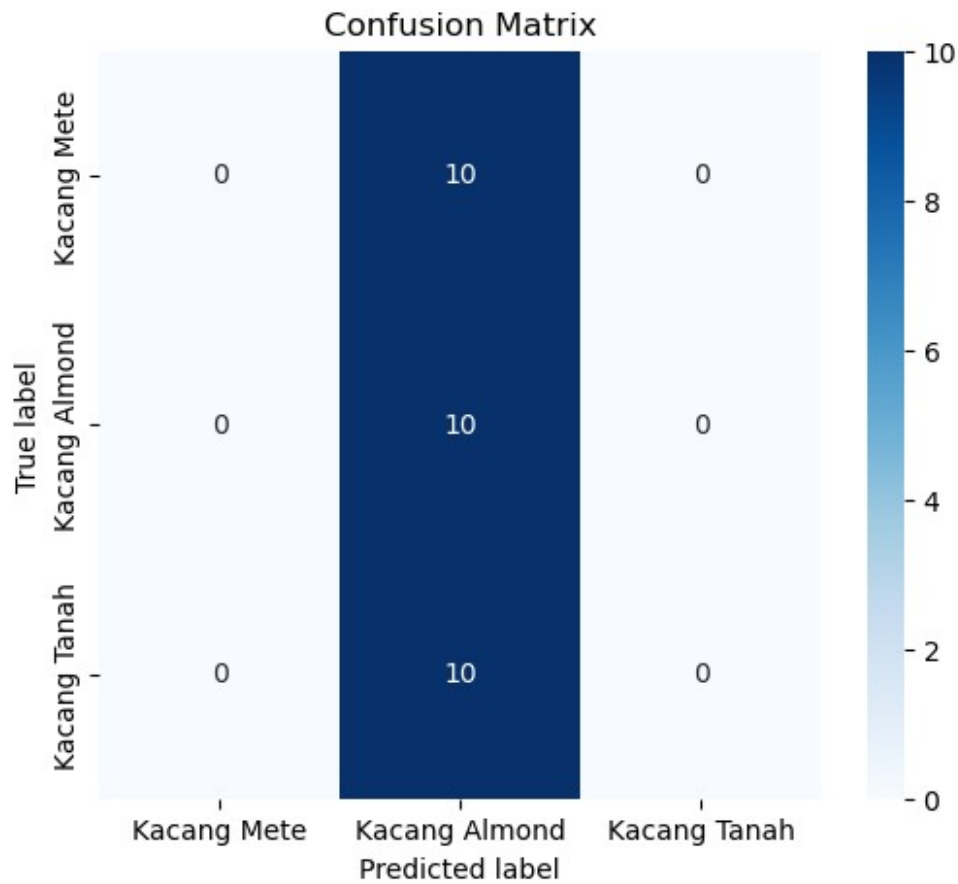
#fmt='d' untuk menampilkan bilangan bulat tanpa desimal
xticklabels=['Kacang Mete', 'Kacang Almond', 'Kacang
Tanah'], yticklabels=['Kacang Mete', 'Kacang Almond', 'Kacang Tanah'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# 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 30 files belonging to 3 classes.
1/1 _____ ls ls/step

```



```
Confusion Matrix:
[[ 0 10  0]
 [ 0 10  0]
 [ 0 10  0]]
Akurasi: 0.3333333333333333
Presisi: [      nan 0.33333333      nan]
Recall: [0.  1.  0.]
F1 Score: [nan 0.5 nan]
```

```

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

data_dir = r"C:\Users\PC\Downloads\CNN\train_data"
data = tf.keras.utils.image_dataset_from_directory(data_dir, seed =
123, image_size=(180,180), batch_size=16)

print(data.class_names)
class_names = data.class_names

Found 300 files belonging to 3 classes.
['Kacang Almond', 'Kacang Mete', 'Kacang Tanah']

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

Found 300 files belonging to 3 classes.

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

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

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

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

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

```

Found 300 files belonging to 3 classes.

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

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

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

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

```
from PIL import Image
import os
```

```
def verify_images(directory):
    invalid_files = []
    for root, _, files in os.walk(directory):
        for file in files:
            if file.lower().endswith(('.jpg', '.jpeg', '.png',
            '.bmp')):
                try:
                    with Image.open(os.path.join(root, file)) as img:
                        img.verify()
                except Exception as e:
                    invalid_files.append(os.path.join(root, file))
    if invalid_files:
        print("Invalid image files:")
        for file in invalid_files:
            print(file)
    else:
        print("All images are valid.")
```

```
base_dir = r'C:\Users\PC\Downloads\CNN\train_data'
verify_images(base_dir)
```

All images are valid.

```
def list_all_files(directory):
    all_files = []
    for root, dirs, files in os.walk(directory):
        for file in files:
            all_files.append(os.path.join(root, file))
    return all_files
```

```
files = list_all_files(base_dir)
```

```
print(f"Total files in dataset: {len(files)}")
for file in files[:10]:
    print(file)
```

Total files in dataset: 300

```
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA1.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA10.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA100.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA11.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA12.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA13.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA14.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA15.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA16.jpg
C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA17.jpg
```

```
from PIL import Image
import os
```

```
def convert_images(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.lower().endswith(('.jpg', '.jpeg', '.png',
            '.bmp')):
                try:
                    image_path = os.path.join(root, file)
                    with Image.open(image_path) as img:
                        new_image_path = image_path.replace(file,
                        file.split('.')[0] + '.jpg')
                        img.convert('RGB').save(new_image_path,
                        'JPEG')
                        print(f"Converted: {image_path} ->
                        {new_image_path}")
                except Exception as e:
                    print(f"Error processing {image_path}: {e}")
```

```
base_dir = r'C:\Users\PC\Downloads\CNN\train_data'
convert_images(base_dir)
```

```
Converted: C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA1.jpg
-> C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA1.jpg
Converted: C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA10.jpg
-> C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA10.jpg
Converted: C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\
KA100.jpg -> C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\
KA100.jpg
Converted: C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA11.jpg
-> C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA11.jpg
Converted: C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA12.jpg
-> C:\Users\PC\Downloads\CNN\train_data\Kacang Almond\KA12.jpg
```


[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Tanah\kacang_tanah_98.jpg
Converted: C:\Users\PC\Downloads\CNN\train_data\Kacang Tanah\
kacang_tanah_99.jpg -> C:\Users\PC\Downloads\CNN\train_data\Kacang
Tanah\kacang_tanah_99.jpg

```
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')
```

Kacang Almond



Kacang Mete



Kacang Almond



Kacang Mete



Kacang Mete



Kacang Almond



Kacang Tanah



Kacang Tanah



Kacang Almond



```
for images, labels in train_ds.take(1):  
    images_array = np.array(images)  
    print(images_array.shape)  
  
(32, 180, 180, 3)  
  
from tensorflow.keras import layers  
from tensorflow.keras.models import Sequential, load_model  
  
Tuner = tf.data.AUTOTUNE  
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size =  
Tuner)
```

```

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

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

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

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

```



```
import tensorflow as tf
import keras

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

from keras._tf_keras.keras.models import load_model

def googlenet(input_shape, n_classes):
```

```

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

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

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

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

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

input = Input(input_shape)

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

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

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

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

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

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

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

model = Model(input, output)
return model
input_shape = 180, 180, 3

```

```
n_classes = 2
K.clear_session()

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

Model: "functional"

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

conv2d_4 (Conv2D)	(None, 22, 22,	18,528	
max_pooling2d_1[...]	96)		
conv2d_6 (Conv2D)	(None, 22, 22,	3,088	
max_pooling2d_1[...]	16)		
max_pooling2d_2	(None, 22, 22,	0	
max_pooling2d_1[...]	192)		
(MaxPooling2D)			
conv2d_3 (Conv2D)	(None, 22, 22,	12,352	
max_pooling2d_1[...]	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[...]	32)		
concatenate	(None, 22, 22,	0	conv2d_3[0]
[0],			
(Concatenate)	256)		conv2d_5[0]
[0],			
[0],			conv2d_7[0]
[0],			

[0]				conv2d_8[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[...	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]				
<hr/>				
max_pooling2d_4	(None, 11, 11,	0		
concatenate_1[0]...	(MaxPooling2D)	480)		
<hr/>				
conv2d_16 (Conv2D)	(None, 11, 11,	46,176		
max_pooling2d_4[...	96)			
<hr/>				
conv2d_18 (Conv2D)	(None, 11, 11,	7,696		
max_pooling2d_4[...	16)			
<hr/>				
max_pooling2d_5	(None, 11, 11,	0		
max_pooling2d_4[...	(MaxPooling2D)	480)		
<hr/>				
conv2d_15 (Conv2D)	(None, 11, 11,	92,352		
max_pooling2d_4[...	192)			
<hr/>				
conv2d_17 (Conv2D)	(None, 11, 11,	179,920	conv2d_16[0]	
[0]	208)			
<hr/>				
conv2d_19 (Conv2D)	(None, 11, 11,	19,248	conv2d_18[0]	
[0]	48)			
<hr/>				
<hr/>				

conv2d_20 (Conv2D)	(None, 11, 11,	30,784	
max_pooling2d_5[0],	64)		
concatenate_2	(None, 11, 11,	0	conv2d_15[0]
(Concatenate)	512)		conv2d_17[0]
			conv2d_19[0]
			conv2d_20[0]
conv2d_22 (Conv2D)	(None, 11, 11,	57,456	
concatenate_2[0]...	112)		
conv2d_24 (Conv2D)	(None, 11, 11,	12,312	
concatenate_2[0]...	24)		
max_pooling2d_6	(None, 11, 11,	0	
concatenate_2[0]...	512)		
(MaxPooling2D)			
conv2d_21 (Conv2D)	(None, 11, 11,	82,080	
concatenate_2[0]...	160)		
conv2d_23 (Conv2D)	(None, 11, 11,	226,016	conv2d_22[0]
[0]	224)		
conv2d_25 (Conv2D)	(None, 11, 11,	38,464	conv2d_24[0]
[0]	64)		

conv2d_26 (Conv2D)	(None, 11, 11,	32,832	
max_pooling2d_6[...]	64)		
concatenate_3	(None, 11, 11,	0	conv2d_21[0]
[0],	(Concatenate)	512)	conv2d_23[0]
[0],			conv2d_25[0]
[0],			conv2d_26[0]
[0]			
conv2d_28 (Conv2D)	(None, 11, 11,	65,664	
concatenate_3[0]...	128)		
conv2d_30 (Conv2D)	(None, 11, 11,	12,312	
concatenate_3[0]...	24)		
max_pooling2d_7	(None, 11, 11,	0	
concatenate_3[0]...	(MaxPooling2D)	512)	
conv2d_27 (Conv2D)	(None, 11, 11,	65,664	
concatenate_3[0]...	128)		
conv2d_29 (Conv2D)	(None, 11, 11,	295,168	conv2d_28[0]
[0]	256)		

conv2d_31 (Conv2D)	(None, 11, 11,	38,464	conv2d_30[0]
[0]	64)		
<hr/>			
conv2d_32 (Conv2D)	(None, 11, 11,	32,832	
max_pooling2d_7[...]	64)		
<hr/>			
concatenate_4	(None, 11, 11,	0	conv2d_27[0]
[0],	(Concatenate)	512)	conv2d_29[0]
[0],			conv2d_31[0]
[0],			conv2d_32[0]
[0]			
<hr/>			
conv2d_34 (Conv2D)	(None, 11, 11,	73,872	
concatenate_4[0]...	144)		
<hr/>			
conv2d_36 (Conv2D)	(None, 11, 11,	16,416	
concatenate_4[0]...	32)		
<hr/>			
max_pooling2d_8	(None, 11, 11,	0	
concatenate_4[0]...	(MaxPooling2D)	512)	
<hr/>			
conv2d_33 (Conv2D)	(None, 11, 11,	57,456	
concatenate_4[0]...	112)		
<hr/>			
conv2d_35 (Conv2D)	(None, 11, 11,	373,536	conv2d_34[0]
[0]	288)		
<hr/>			

conv2d_37 (Conv2D)	(None, 11, 11, 64)	51,264	conv2d_36[0]
conv2d_38 (Conv2D)	(None, 11, 11, 64)	32,832	
max_pooling2d_8[...]			
concatenate_5	(None, 11, 11, 528)	0	conv2d_33[0]
[0],			conv2d_35[0]
(Concatenate)			conv2d_37[0]
[0],			conv2d_38[0]
[0],			
[0]			
conv2d_40 (Conv2D)	(None, 11, 11, 160)	84,640	
concatenate_5[0]...			
conv2d_42 (Conv2D)	(None, 11, 11, 32)	16,928	
concatenate_5[0]...			
max_pooling2d_9	(None, 11, 11, 528)	0	
concatenate_5[0]...			
(MaxPooling2D)			
conv2d_39 (Conv2D)	(None, 11, 11, 256)	135,424	
concatenate_5[0]...			

conv2d_41 (Conv2D)	(None, 11, 11,	461,120	conv2d_40[0]
[0]	320)		
conv2d_43 (Conv2D)	(None, 11, 11,	102,528	conv2d_42[0]
[0]	128)		
conv2d_44 (Conv2D)	(None, 11, 11,	67,712	
max_pooling2d_9[...	128)		
concatenate_6	(None, 11, 11,	0	conv2d_39[0]
[0],	(Concatenate)	832)	conv2d_41[0]
[0],			conv2d_43[0]
[0],			conv2d_44[0]
[0]			
max_pooling2d_10	(None, 6, 6, 832)	0	
concatenate_6[0]...			
(MaxPooling2D)			
conv2d_46 (Conv2D)	(None, 6, 6, 160)	133,280	
max_pooling2d_10...			
conv2d_48 (Conv2D)	(None, 6, 6, 32)	26,656	
max_pooling2d_10...			
max_pooling2d_11	(None, 6, 6, 832)	0	
max_pooling2d_10...			
(MaxPooling2D)			
conv2d_45 (Conv2D)	(None, 6, 6, 256)	213,248	

max_pooling2d_10...			
conv2d_47 (Conv2D)	(None, 6, 6, 320)	461,120	conv2d_46[0]
conv2d_49 (Conv2D)	(None, 6, 6, 128)	102,528	conv2d_48[0]
conv2d_50 (Conv2D)	(None, 6, 6, 128)	106,624	
max_pooling2d_11...			
concatenate_7	(None, 6, 6, 832)	0	conv2d_45[0]
(Concatenate)			conv2d_47[0]
			conv2d_49[0]
			conv2d_50[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]
conv2d_55 (Conv2D)	(None, 6, 6, 128)	153,728	conv2d_54[0]

conv2d_56 (Conv2D)	(None, 6, 6, 128)	106,624	
max_pooling2d_12...			
concatenate_8	(None, 6, 6,	0	conv2d_51[0]
[0],			
(Concatenate)	1024)		conv2d_53[0]
[0],			
			conv2d_55[0]
[0],			
			conv2d_56[0]
[0]			
average_pooling2d	(None, 4, 4,	0	
concatenate_8[0]...			
(AveragePooling2D)	1024)		
dropout (Dropout)	(None, 4, 4,	0	
average_pooling2...			
	1024)		
flatten (Flatten)	(None, 16384)	0	dropout[0][0]
dense (Dense)	(None, 2)	32,770	flatten[0][0]

Total params: 6,006,322 (22.91 MB)

Trainable params: 6,006,322 (22.91 MB)

Non-trainable params: 0 (0.00 B)

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
import numpy as np

model = Sequential([

```



```

        Flatten(input_shape=(180, 180, 3)),
        Dense(128, activation='relu'),
        Dense(3, activation='softmax')
    ])

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

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

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

```

Epoch 1/30

```

c:\Users\PC\anaconda3\Lib\site-packages\keras\src\layers\reshaping\
flatten.py:37: 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)

```

```

8/8 _____ 1s 83ms/step - accuracy: 0.3185 - loss:
15332.8311 - val_accuracy: 0.4091 - val_loss: 609.9547

```

Epoch 2/30

```

8/8 _____ 0s 60ms/step - accuracy: 0.3204 - loss:
3997.1213 - val_accuracy: 0.4091 - val_loss: 1565.5082

```

Epoch 3/30

```

8/8 _____ 1s 64ms/step - accuracy: 0.4285 - loss:
1844.3956 - val_accuracy: 0.5682 - val_loss: 622.5387

```

Epoch 4/30

```

8/8 _____ 0s 59ms/step - accuracy: 0.5320 - loss:
1156.9677 - val_accuracy: 0.5227 - val_loss: 1942.3004

```

Epoch 5/30

```

8/8 _____ 0s 61ms/step - accuracy: 0.5856 - loss:
1437.0381 - val_accuracy: 0.4773 - val_loss: 992.5567

```

Epoch 6/30

```

8/8 _____ 0s 60ms/step - accuracy: 0.6439 - loss:
902.8458 - val_accuracy: 0.7955 - val_loss: 344.0068

```

Epoch 7/30

```

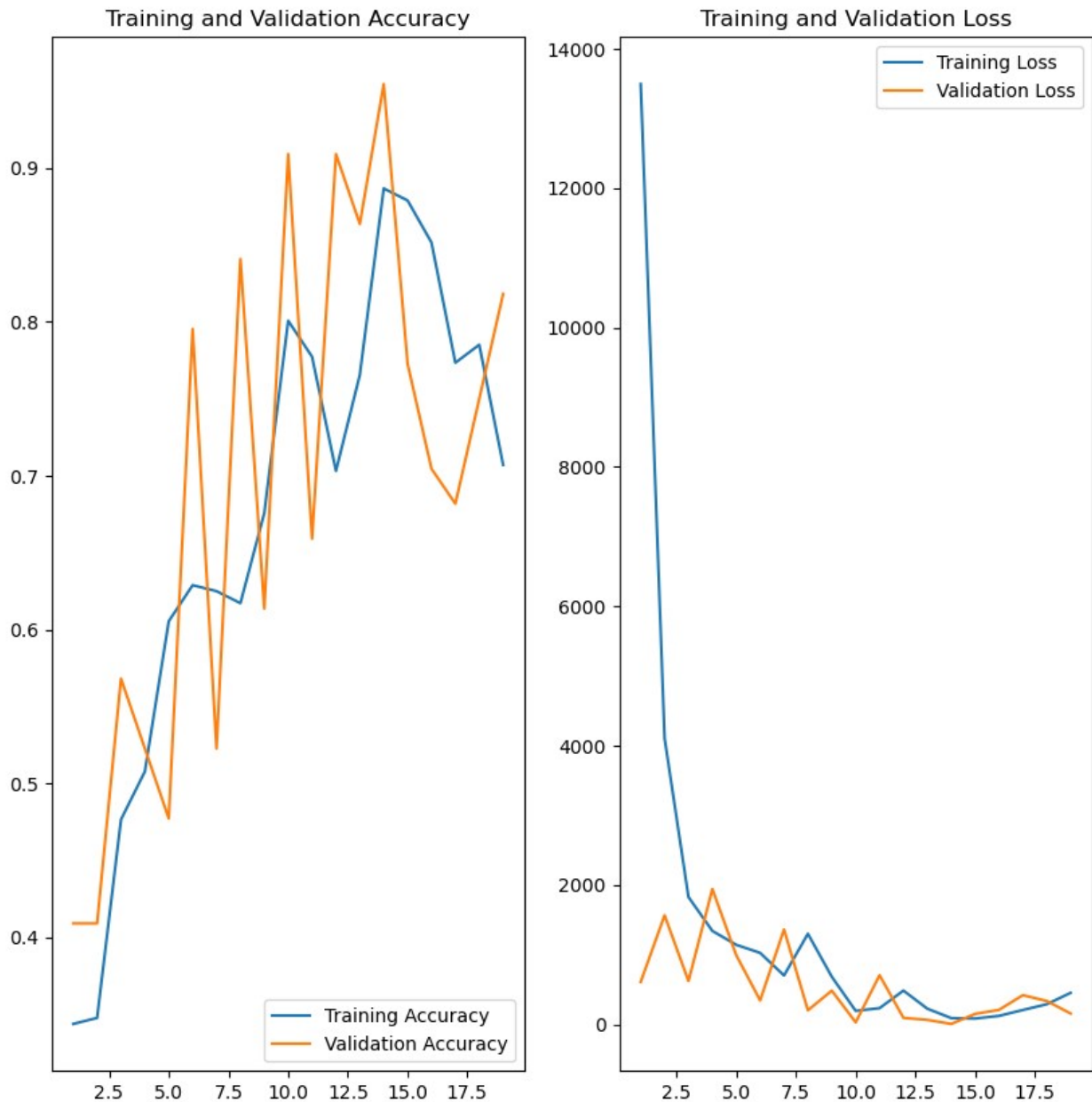
8/8 _____ 0s 60ms/step - accuracy: 0.6502 - loss:
638.4806 - val_accuracy: 0.5227 - val_loss: 1361.4243
Epoch 8/30
8/8 _____ 0s 59ms/step - accuracy: 0.6297 - loss:
1086.7289 - val_accuracy: 0.8409 - val_loss: 201.7935
Epoch 9/30
8/8 _____ 0s 60ms/step - accuracy: 0.6831 - loss:
649.0095 - val_accuracy: 0.6136 - val_loss: 484.1543
Epoch 10/30
8/8 _____ 1s 60ms/step - accuracy: 0.7526 - loss:
299.6664 - val_accuracy: 0.9091 - val_loss: 29.2222
Epoch 11/30
8/8 _____ 0s 59ms/step - accuracy: 0.8405 - loss:
119.4641 - val_accuracy: 0.6591 - val_loss: 704.9731
Epoch 12/30
8/8 _____ 0s 61ms/step - accuracy: 0.6857 - loss:
587.0234 - val_accuracy: 0.9091 - val_loss: 93.5263
Epoch 13/30
8/8 _____ 1s 68ms/step - accuracy: 0.7678 - loss:
206.0521 - val_accuracy: 0.8636 - val_loss: 64.5135
Epoch 14/30
8/8 _____ 1s 68ms/step - accuracy: 0.8938 - loss:
60.1409 - val_accuracy: 0.9545 - val_loss: 6.2766
Epoch 15/30
8/8 _____ 1s 66ms/step - accuracy: 0.9077 - loss:
65.8529 - val_accuracy: 0.7727 - val_loss: 149.9149
Epoch 16/30
8/8 _____ 1s 63ms/step - accuracy: 0.8522 - loss:
103.9424 - val_accuracy: 0.7045 - val_loss: 207.9891
Epoch 17/30
8/8 _____ 1s 69ms/step - accuracy: 0.7458 - loss:
241.8094 - val_accuracy: 0.6818 - val_loss: 418.9024
Epoch 18/30
8/8 _____ 0s 61ms/step - accuracy: 0.7655 - loss:
371.1157 - val_accuracy: 0.7500 - val_loss: 335.6447
Epoch 19/30
8/8 _____ 1s 64ms/step - accuracy: 0.7259 - loss:
422.9583 - val_accuracy: 0.8182 - val_loss: 155.6053

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

plt.subplot(1, 2, 2)

```

```
plt.plot(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('gugelnet.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format

is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.

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

model = load_model(r'gugenet.h5')
class_names = ['Kacang Almond', 'Kacang Mete', 'Kacang Tanah']

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

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

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

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

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

result = classify_images(r'C:\Users\PC\Downloads\CNN\test_data\
Validation Kacang Mete/image 3.jpg', save_path='image 3.jpg')
print(result)
```

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

1/1 _____ 0s 144ms/step

Prediksi: Kacang Mete

Confidence: 57.61%

Prediksi: Kacang Mete dengan confidence 57.61%. Gambar asli disimpan di image 3.jpg.

```

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

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'C:\Users\PC\Downloads\CNN\test_data',
    labels='inferred',
    label_mode='categorical',
    batch_size=32,
    image_size=(180, 180)
)

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

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

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

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

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

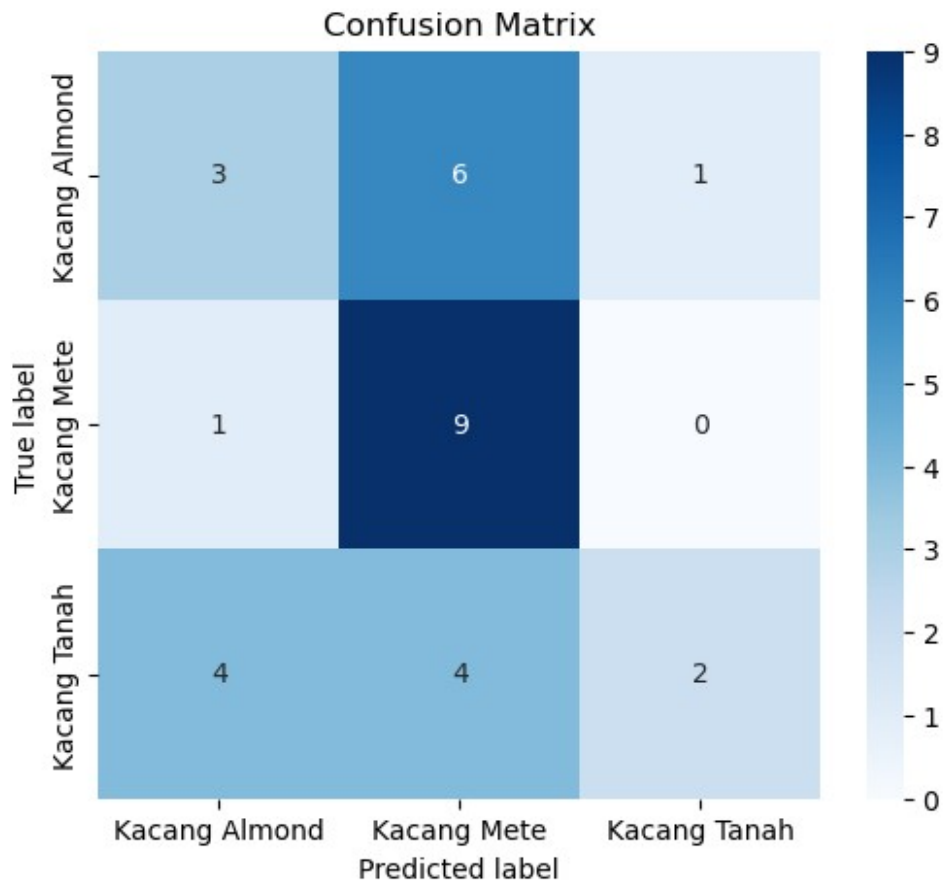
f1_score = 2 * (precision * recall) / (precision + recall)

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

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

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

```



Confusion Matrix:

```
[[3 6 1]
```

```
[1 9 0]
```

```
[4 4 2]]
```

Akurasi: 0.4666666666666667

Presisi: [0.375 0.47368421 0.66666667]

Recall: [0.3 0.9 0.2]

F1 Score: [0.33333333 0.62068966 0.30769231]

```

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

# Load model yang sudah dilatih
model = load_model(r'd:\Matakuliah Semester 5\Pembelajaran Mesin dan Pembelajaran Mendalam\UAS\Googlenet_A_Numpy_RAFI\gugelnet.h5')

# Nama kelas yang sesuai dengan 3 jenis kacang
class_names = ['Kacang Tanah', 'Kacang Mete', 'Kacang Almond']

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

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

        # Mendapatkan kelas dengan probabilitas tertinggi
        class_idx = np.argmax(result)
        confidence_scores = result.numpy()

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

def custom_progress_bar(confidence, color1, color2, color3):
    percentage1 = confidence[0] * 100
    percentage2 = confidence[1] * 100
    percentage3 = confidence[2] * 100
    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}% Kacang Tanah
  </div>
  <div style="width: {percentage2:.2f}%; background: {color2}; color: white; text-align: center; height: 24px; float: left;">
    {percentage2:.2f}% Kacang Mete
  </div>
  <div style="width: {percentage3:.2f}%; background: {color3}; color: white; text-align: center; height: 24px; float: left;">
    {percentage3:.2f}% Kacang Almond
  </div>
</div>
"""
    st.sidebar.markdown(progress_html, unsafe_allow_html=True)

st.title("Prediksi Klasifikasi Jenis Kacang - Numpy")

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":
                primary_color = "#007BFF"
                secondary_color = "#FF4136"
                tertiary_color = "#4CAF50"
                label_color = primary_color if label == "Kacang Tanah" else secondary_color if label == "Kacang Mete" else tertiary_color

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

                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.")

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)

```