Tugas Modul Convolutional Neural Network

MobileNet

Import Library

- Tahap pertama adalah import seluruh library yang dibutuhkan

```
#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

#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
```

Load Data

- Load dataset berdasarkan path dimana dataset disimpan

```
count = 0 #digunakan untuk menghitung jumlah gambar
dirs = os.listdir(r'D:\semester 5\Mesin Learning\dataset\train_data')
for dir in dirs:
    files = list(os.listdir(r'D:\semester 5\Mesin Learning\dataset\
train_data/'+dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Fuji Apple Folder has 80 Images
Golden Delicious Apple Folder has 80 Images
Granny Smith Apple Folder has 80 Images
Images Folder has 240 Images
```

Load Images into Arrays as Dataset

- Membuat dataset dari gambar yang ada di direktori

```
# Parameter
base_dir = r'D:\semester 5\Mesin Learning\dataset\train_data'
#direktori folder dataset
img_size = 180 #mengubah ukuran gambar menjadi 180
batch = 32 #jumlah sample (gambar) yang akan diproses pada satu kali
iterasi
validation_split = 0.1 #data pelatihan yang akan digunakan sebagai
data validasi
```

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

```
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir, #path direktori, subfolder dianggap sebagai label
    seed=123, #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 240 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)

Class Names: ['Fuji Apple', 'Golden Delicious Apple', 'Granny Smith
Apple']
```

Train-Validation-Test Split

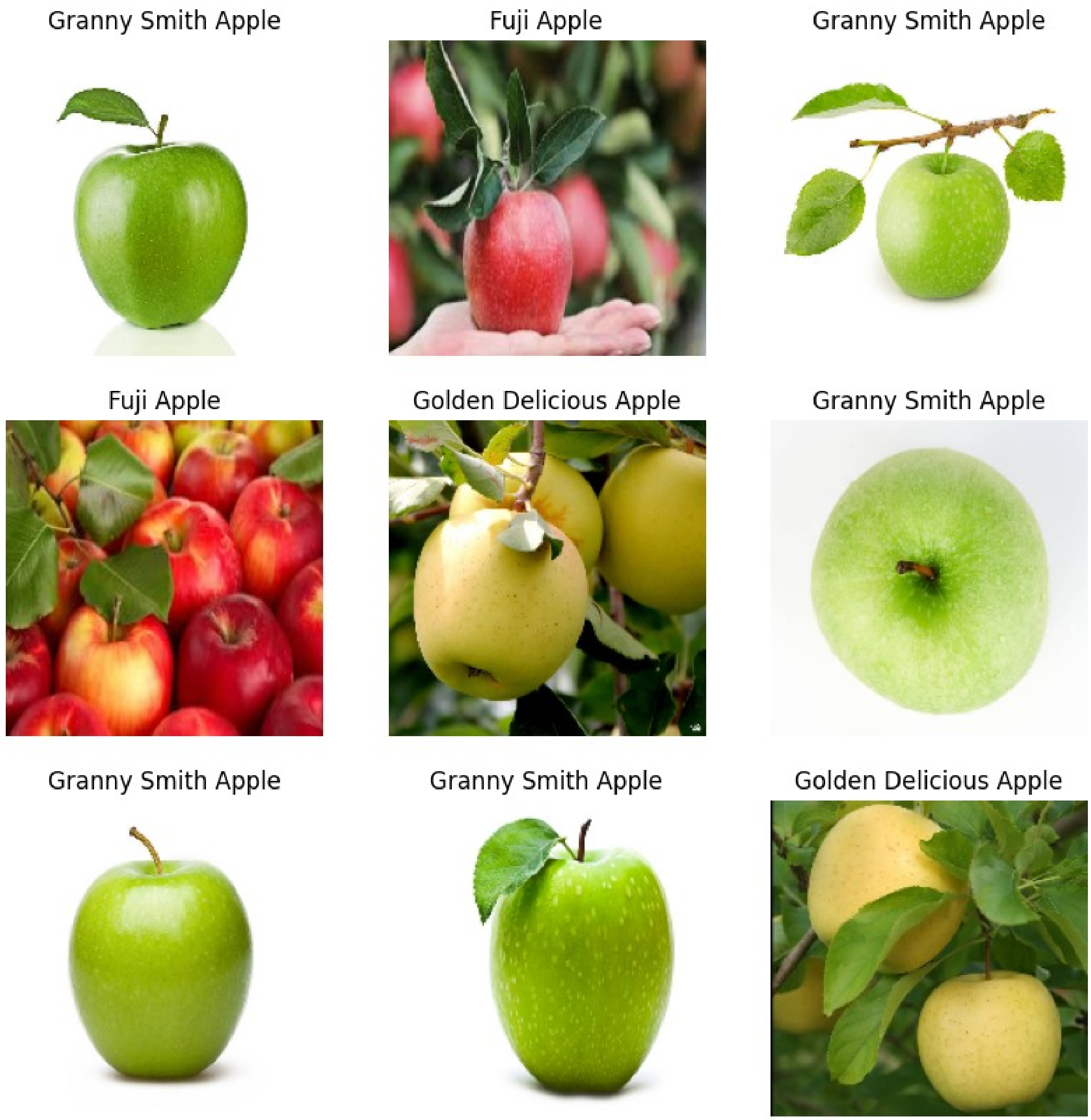
- Membagi dataset menjadi tiga subset yaitu train, validation, dan test
  - Train, digunakan untuk melatih model agar mengenali pola dalam data
  - Validation, digunakan untuk mengevaluasi performa model selama pelatihan
  - Test, digunakan untuk menguji model setelah pelatihan

```
#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)
```

```
Total Images: 8
Train Images: 8
Validation Images: 0
```

```
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

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
```




```
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)
```

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

#Mengatur AUTOTUNE untuk pemrosesan data otomatis oleh tensorflow
#AUTOTUNE digunakan untuk memungkinkan tensorflow mengoptimalkan
jumlah thread secara otomatis saat memproses data
AUTOTUNE = tf.data.AUTOTUNE

#mengoptimalkan dataset pelatihan (train_ds)
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size =
AUTOTUNE)
#cache digunakan untuk menyimpan dataser di memori agar lebih cepat
diakses
#shuffle mengacak data dalam batch agar model tidak terlalu terlatih
pada urutan tertentu
#prefetch untuk menyiapkan data batch berikutnya secara otomatis

#mengoptimalkan dataset validasi (val_ds)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
```

Data Augmentation

- Digunakan untuk menambah variasi data pelatihan dengan membuat gambar baru dari yang sudah ada seperti dengan rotasi, flipping, zooming, dan sebagainy
- Untuk mengurangi overfitting dan memperbesar dataset tanpa mengumpulkan data baru

```
data_augmentation = Sequential([
    layers.RandomFlip("diagonal", 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%
])

#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')
```

MobileNet

- Salah satu algoritma yang dirancang untuk perangkat dengan keterbatasan sumber daya seperti smartphone

```
#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
```

```python
#include_top=False berarti tidak menggunakan lapisan klasifikasi dari mobilenet hanya bagian ekstraksi fitur
base_model = MobileNet(include_top=False, input_shape=(img_size, img_size, 3))

#membuka (unfreeze beberapa lapisan untuk proses fine tuning)
base_model.trainable = True #seluruh model bisa dilatih
fine_tune_at = len(base_model.layers) // 2  #menentukan bahwa setengah lapisan terakhir akan di unfreeze
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False #mengunci (freeze) lapisan pertama hingga setengah bagian pertama agar tidak dilatih kembali

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

```
C:\Users\HP\AppData\Local\Temp\ipykernel_192\1409442124.py:7: UserWarning: `input_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
  base_model = MobileNet(include_top=False, input_shape=(img_size, img_size, 3))
```

```python
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
)

#menampilkan ringkasan dari model
model.summary()
```

```
Model: "sequential_5"

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

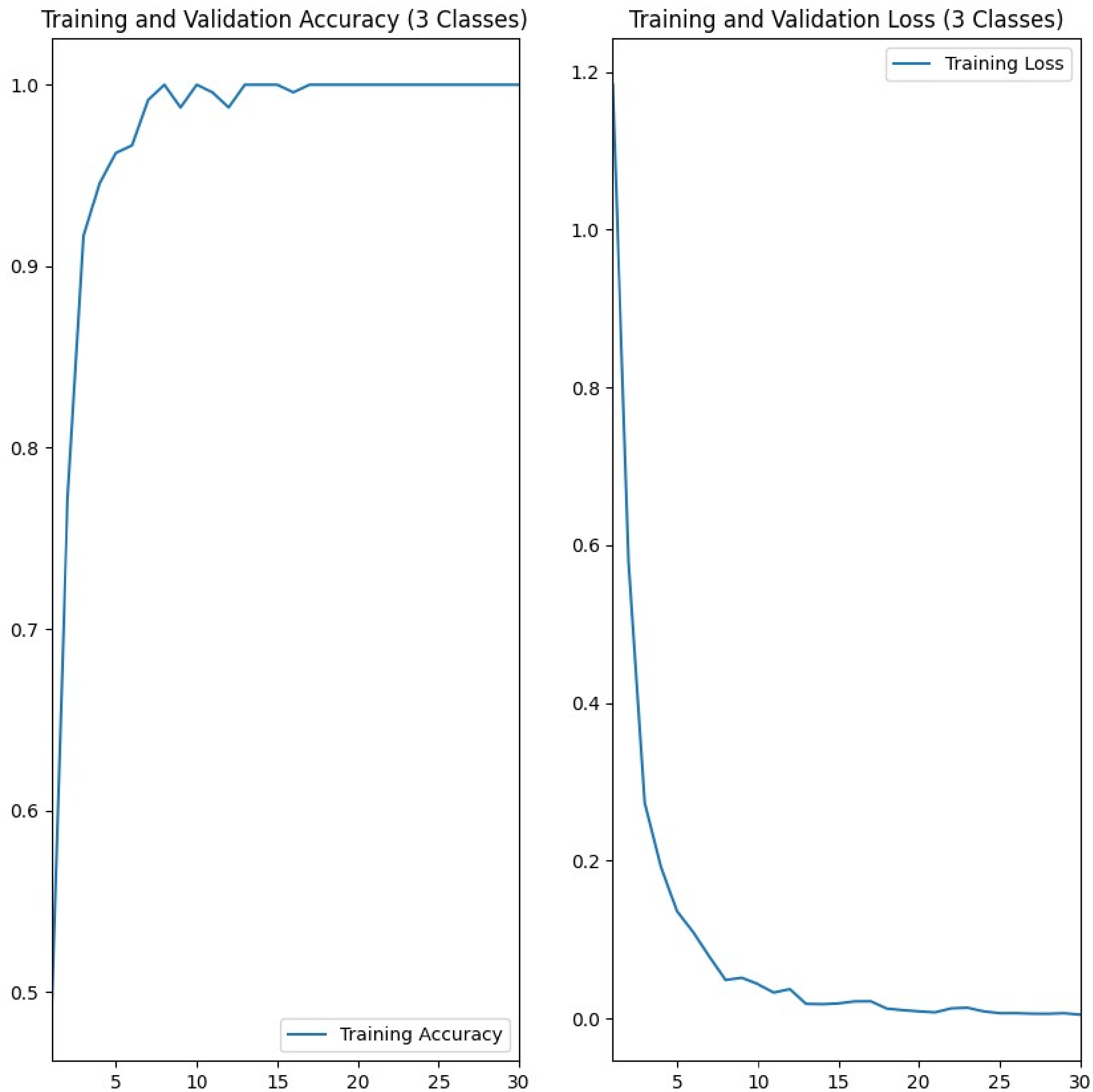
```
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━━━━━━━━━━━━━━━━╋━━━━━━━━━━
━━━━━┩
│ sequential_4 (Sequential)        │ (None, 180, 180, 3)     │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ rescaling_2 (Rescaling)          │ (None, 180, 180, 3)     │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ mobilenet_1.00_224 (Functional) │ (None, 5, 5, 1024)      │
3,228,864 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ global_average_pooling2d_2       │ (None, 1024)            │
0 │
│ (GlobalAveragePooling2D)         │                         │
│
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ dense_4 (Dense)                  │ (None, 128)             │
131,200 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ dropout_2 (Dropout)              │ (None, 128)             │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ dense_5 (Dense)                  │ (None, 3)               │
387 │
└──────────────────────────────────┴─────────────────────────┴──────────
─────┘

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

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

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

#early stopping digunakan untuk menghentikan pelatihan lebih awal jika
model tidak ada peningkatan
from tensorflow.keras.callbacks import EarlyStopping

#Ada fungsi early stopping
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
8/8 ━━━━━━━━━━━━━━━━━━━━ 14s 682ms/step - accuracy: 0.4598 - loss:
1.2696
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 551ms/step - accuracy: 0.7496 - loss:
0.5958
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 503ms/step - accuracy: 0.9285 - loss:
0.2618
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 517ms/step - accuracy: 0.9700 - loss:
0.1722
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 539ms/step - accuracy: 0.9609 - loss:
0.1444
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 556ms/step - accuracy: 0.9660 - loss:
0.1179
Epoch 7/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 504ms/step - accuracy: 0.9937 - loss:
0.0815
Epoch 8/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 498ms/step - accuracy: 1.0000 - loss:
0.0495
Epoch 9/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 493ms/step - accuracy: 0.9905 - loss:
0.0540
Epoch 10/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 528ms/step - accuracy: 1.0000 - loss:
0.0337
Epoch 11/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 603ms/step - accuracy: 0.9979 - loss:
0.0344
Epoch 12/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 514ms/step - accuracy: 0.9861 - loss:
0.0393
Epoch 13/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 571ms/step - accuracy: 1.0000 - loss:
0.0200
Epoch 14/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 5s 572ms/step - accuracy: 1.0000 - loss:
0.0185
Epoch 15/30
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 551ms/step - accuracy: 1.0000 - loss:
0.0215
Epoch 16/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 573ms/step - accuracy: 0.9979 - loss:
0.0180
Epoch 17/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 531ms/step - accuracy: 1.0000 - loss:
0.0220
Epoch 18/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 509ms/step - accuracy: 1.0000 - loss:
0.0140
Epoch 19/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 482ms/step - accuracy: 1.0000 - loss:
0.0099
Epoch 20/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 520ms/step - accuracy: 1.0000 - loss:
0.0077
Epoch 21/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 479ms/step - accuracy: 1.0000 - loss:
0.0072
Epoch 22/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 478ms/step - accuracy: 1.0000 - loss:
0.0101
Epoch 23/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 505ms/step - accuracy: 1.0000 - loss:
0.0162
Epoch 24/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 518ms/step - accuracy: 1.0000 - loss:
0.0094
Epoch 25/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 527ms/step - accuracy: 1.0000 - loss:
0.0054
Epoch 26/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 516ms/step - accuracy: 1.0000 - loss:
0.0062
Epoch 27/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 503ms/step - accuracy: 1.0000 - loss:
0.0059
Epoch 28/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 485ms/step - accuracy: 1.0000 - loss:
0.0054
Epoch 29/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 519ms/step - accuracy: 1.0000 - loss:
0.0054
Epoch 30/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 476ms/step - accuracy: 1.0000 - loss:
0.0038

# Memeriksa apakah 'val_accuracy' tersedia
val_acc = 'val_accuracy' if 'val_accuracy' in history.history else
```

```
None
val_loss = 'val_loss' if 'val_loss' in history.history else None

# Membuat range untuk epoch berdasarkan panjang data loss dari
pelatihan
epochs_range = range(1, len(history.history['loss']) + 1)

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

# Grafik pertama (Training and Validation Accuracy)
plt.subplot(1, 2, 1)
plt.plot(epochs_range, history.history['accuracy'], label='Training
Accuracy')
if val_acc:
    plt.plot(epochs_range, history.history[val_acc], label='Validation
Accuracy')
plt.legend(loc='lower right')
plt.xlim(1, len(epochs_range))
plt.title('Training and Validation Accuracy (3 Classes)')

# Grafik kedua (Training and Validation Loss)
plt.subplot(1, 2, 2)
plt.plot(epochs_range, history.history['loss'], label='Training Loss')
if val_loss:
    plt.plot(epochs_range, history.history[val_loss],
label='Validation Loss')
plt.legend(loc='upper right')
plt.xlim(1, len(epochs_range))
plt.title('Training and Validation Loss (3 Classes)')

plt.show()
```




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

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

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

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

#memuat model yang sudah dilatih
model = load_model(r'D:\semester 5\Mesin Learning\dataset\
model_mobilenet.h5')
class_names = ['Fuji Apple', 'Golden Delicious Apple', 'Granny Smith
Apple'] #kelas yang ada pada model

#fungsi untuk mengklasifikasikan gambar dan menyimpan gambar asli
def classify_images(image_path, save_path='predicted_image.png'):
    try:
        #memuat dan mempersiapkan gambar untuk prediksi
        input_image = tf.keras.utils.load_img(image_path,
target_size=(180, 180)) #membuat gambar dari path dan mnegubah
ukurannya menjadi 180x180 pixel
        input_image_array = tf.keras.utils.img_to_array(input_image)
#mengubah gambar jadi array numpy agar bisa di proses model
        input_image_exp_dim = tf.expand_dims(input_image_array, 0)
#menambahkan dimensi batch agar sesuai dengan input model

#dimensi menjadi (1, 180, 180, 3)

        #melakukan prediksi
        predictions = model.predict(input_image_exp_dim) #melakukan
prediksi pada gambar yang telah diproses
        result = tf.nn.softmax(predictions[0]) #menghitung hasil
prediksi menggunakan softmax untuk mendapatkan probabilitas tiap kelas
        class_idx = np.argmax(result) #menemukan indeks kelas dengan
probabilitas tertinggi
        confidence = np.max(result) * 100 #menghitung confidence dalam
persentase

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

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

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

```
#contoh penggunaan fungsi
result = classify_images(r'D:\semester 5\Mesin Learning\dataset\
test_data\Fuji Apple\Fuji087.png', save_path='Fuji Apple.png')
print(result)
```

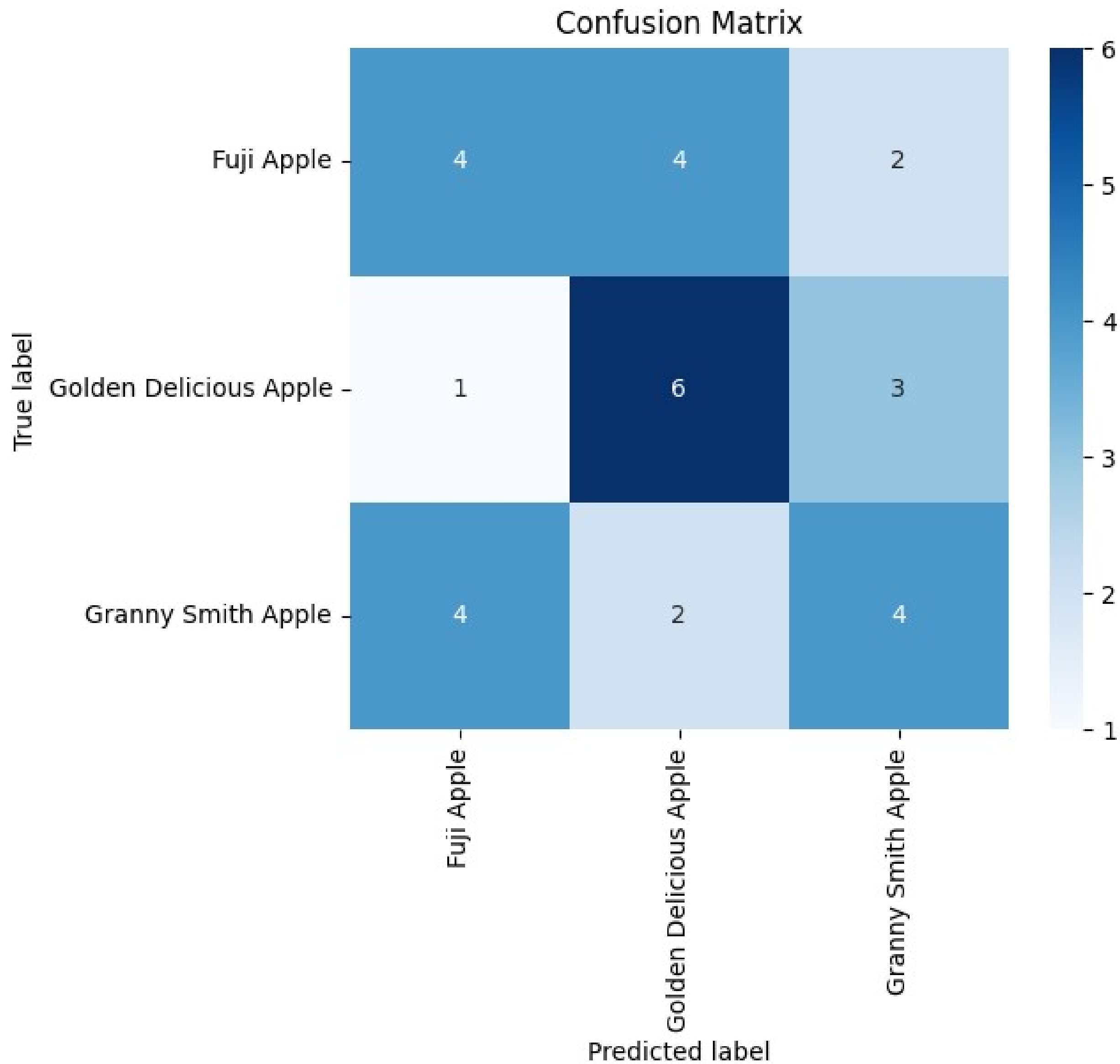
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 478ms/step
Prediksi: Fuji Apple
Confidence: 40.55%
Prediksi: Fuji Apple dengan confidence 40.55%. Gambar asli disimpan di
Fuji Apple.png.
```

```
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:\semester 5\Mesin Learning\dataset\
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)

#mnghitung 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=["Fuji Apple", "Golden Delicious Apple",
"Granny Smith Apple"], yticklabels=["Fuji Apple", "Golden Delicious
Apple", "Granny Smith Apple"])
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 ━━━━━━━━━━━━━━━━━━━━ 1s 744ms/step
```



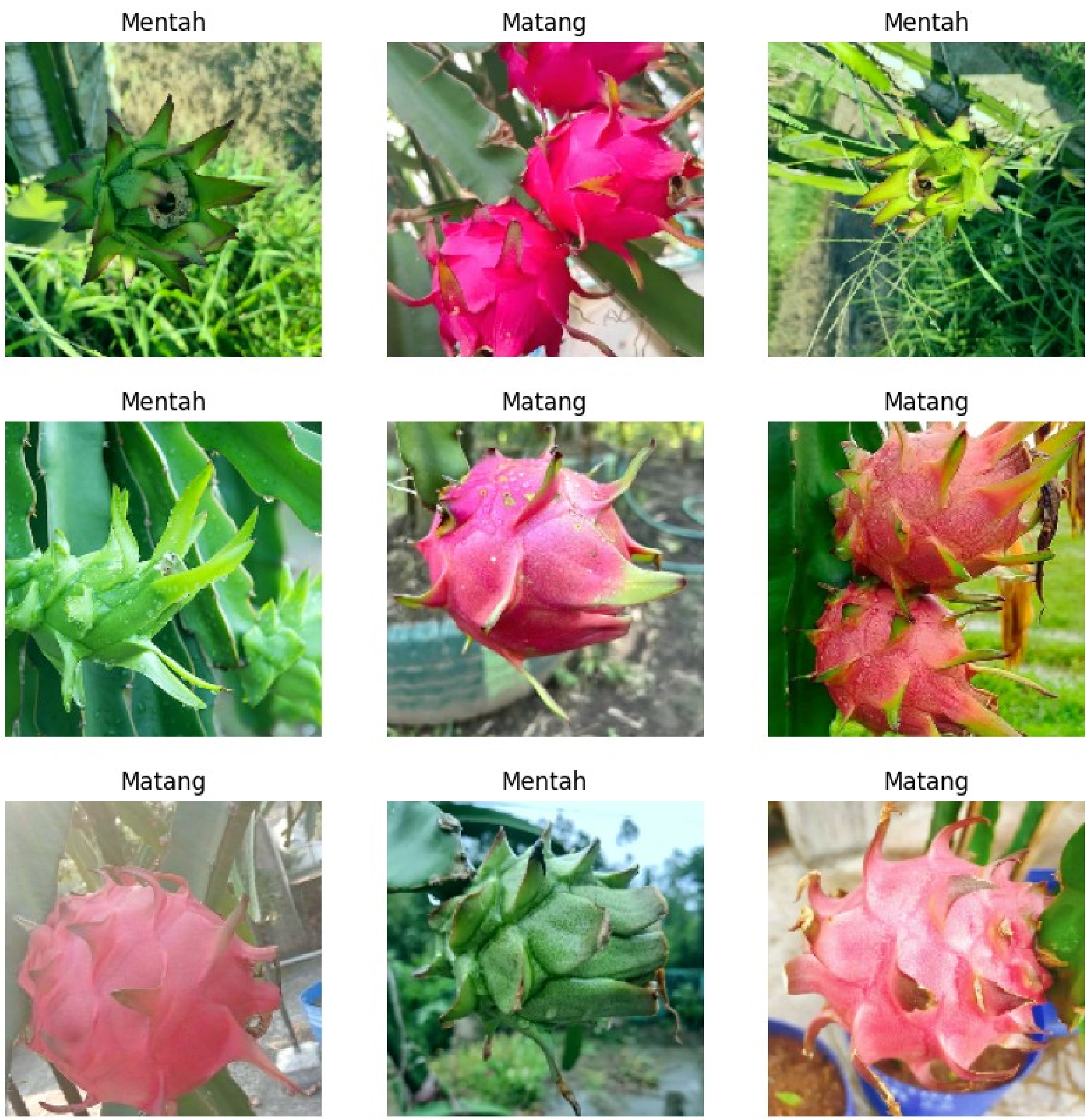

```
Confusion Matrix:
 [[4 4 2]
 [1 6 3]
 [4 2 4]]
Akurasi: 0.4666666666666667
Presisi: [0.44444444 0.5        0.44444444]
Recall: [0.4 0.6 0.4]
F1 Score: [0.42105263 0.54545455 0.42105263]
```

```python
import tensorflow as tf
import numpy as np
from matplotlib import pyplot as plt
#load data
data_dir = r"D:\semester 5\Mesin Learning\Tugas6_B_11978\train_data"
#Randomize data yang telah di load sekaligus resize menjadi 180 x 180
data = tf.keras.utils.image_dataset_from_directory(data_dir, seed=123,
image_size=(180, 180), batch_size=16)
print(data.class_names)

class_names = data.class_names

img_size = 180
batch = 32
validation_split = 0.1

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

total_count = len(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)
```

```
Found 1600 files belonging to 2 classes.
['Matang', 'Mentah']
Found 1600 files belonging to 2 classes.
Total Images: 50
Train Images: 45
Validation Images: 5
```

```python
import matplotlib.pyplot as plt

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

#tampilkan untuk memastikan data sudah di load
for images, labels in train_ds.take(1):
    for i in range(9):
        plt.subplot(3,3, i+1)
```

```
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(class_names [labels[i]])
        plt.axis('off')
```




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

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

```
(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)

#Augmentasi data dengan menggunakan Sequential
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape = (img_size,
img_size,3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])

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

C:\Users\HP\AppData\Roaming\Python\Python312\site-packages\keras\src\
layers\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)
```

```python
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

#membuat model from scratch
```

```python
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
```

```python
#Pastikan input shae dan jumlah kelas sesuai
input_shape = 180, 180, 3
n_classes = 2

#Clear Cache Keras menggunakan clear session
K.clear_session()
#buat model dengan
model = googlenet(input_shape, n_classes)
model.summary()
```

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


Model: "functional"
```

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

```
│                         │ 192)                  │                │
│
├─────────────────────────┼───────────────────────┼────────────────┼─────────────────
──────┤
│ max_pooling2d_1         │ (None, 22, 22,        │              0 │ conv2d_2[0]
[0]       │
│ (MaxPooling2D)          │ 192)                  │                │
│
├─────────────────────────┼───────────────────────┼────────────────┼─────────────────
──────┤
│ 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[… │
│ (MaxPooling2D)          │ 192)                  │                │
│
├─────────────────────────┼───────────────────────┼────────────────┼─────────────────
──────┤
│ 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],    │
│                           │                        │               │ conv2d_7[0]
[0],    │
│                           │                        │               │ conv2d_8[0]
[0]     │
├───────────────────────────┼────────────────────────┼───────────────┼─────────────────
───────┤
│ conv2d_10 (Conv2D)        │ (None, 22, 22,         │        32,896 │
concatenate[0][0] │
│                           │ 128)                   │               │
│
├───────────────────────────┼────────────────────────┼───────────────┼─────────────────
───────┤
│ conv2d_12 (Conv2D)        │ (None, 22, 22,         │         8,224 │
concatenate[0][0] │
│                           │ 32)                    │               │
│
├───────────────────────────┼────────────────────────┼───────────────┼─────────────────
───────┤
│ max_pooling2d_3           │ (None, 22, 22,         │             0 │
concatenate[0][0] │
│ (MaxPooling2D)            │ 256)                   │               │
│
├───────────────────────────┼────────────────────────┼───────────────┼─────────────────
───────┤
│ conv2d_9 (Conv2D)         │ (None, 22, 22,         │        32,896 │
concatenate[0][0] │
│                           │ 128)                   │               │
│
├───────────────────────────┼────────────────────────┼───────────────┼─────────────────
───────┤
│ conv2d_11 (Conv2D)        │ (None, 22, 22,         │       221,376 │ conv2d_10[0]
[0]    │
│                           │ 192)                   │               │
│
├───────────────────────────┼────────────────────────┼───────────────┼─────────────────
───────┤
│ conv2d_13 (Conv2D)        │ (None, 22, 22,         │        76,896 │ conv2d_12[0]
[0]    │
│                           │ 96)                    │               │
│
├───────────────────────────┼────────────────────────┼───────────────┼─────────────────
───────┤
```

```
│ conv2d_14 (Conv2D)  │ (None, 22, 22,    │     16,448 │
max_pooling2d_3[… │
│                     │ 64)               │            │
│
├─────────────────────┼───────────────────┼────────────┼───────────────
──────┤
│ concatenate_1       │ (None, 22, 22,    │          0 │ conv2d_9[0]
[0],    │
│ (Concatenate)       │ 480)              │            │ conv2d_11[0]
[0],   │
│                     │                   │            │ conv2d_13[0]
[0],   │
│                     │                   │            │ conv2d_14[0]
[0]    │
├─────────────────────┼───────────────────┼────────────┼───────────────
──────┤
│ max_pooling2d_4     │ (None, 11, 11,    │          0 │
concatenate_1[0]… │
│ (MaxPooling2D)      │ 480)              │            │
│
├─────────────────────┼───────────────────┼────────────┼───────────────
──────┤
│ conv2d_16 (Conv2D)  │ (None, 11, 11,    │     46,176 │
max_pooling2d_4[… │
│                     │ 96)               │            │
│
├─────────────────────┼───────────────────┼────────────┼───────────────
──────┤
│ conv2d_18 (Conv2D)  │ (None, 11, 11,    │      7,696 │
max_pooling2d_4[… │
│                     │ 16)               │            │
│
├─────────────────────┼───────────────────┼────────────┼───────────────
──────┤
│ max_pooling2d_5     │ (None, 11, 11,    │          0 │
max_pooling2d_4[… │
│ (MaxPooling2D)      │ 480)              │            │
│
├─────────────────────┼───────────────────┼────────────┼───────────────
──────┤
│ conv2d_15 (Conv2D)  │ (None, 11, 11,    │     92,352 │
max_pooling2d_4[… │
│                     │ 192)              │            │
│
├─────────────────────┼───────────────────┼────────────┼───────────────
──────┤
│ conv2d_17 (Conv2D)  │ (None, 11, 11,    │    179,920 │ conv2d_16[0]
[0]    │
│                     │ 208)              │            │
```

```
│
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────
───────┤
│ conv2d_19 (Conv2D)        │ (None, 11, 11,         │         19,248 │ conv2d_18[0]
[0]    │
│                           │ 48)                    │                │
│
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────
───────┤
│ conv2d_20 (Conv2D)        │ (None, 11, 11,         │         30,784 │
max_pooling2d_5[… │
│                           │ 64)                    │                │
│
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────
───────┤
│ concatenate_2             │ (None, 11, 11,         │              0 │ conv2d_15[0]
[0],   │
│ (Concatenate)             │ 512)                   │                │ conv2d_17[0]
[0],   │
│                           │                        │                │ conv2d_19[0]
[0],   │
│                           │                        │                │ conv2d_20[0]
[0]    │
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────
───────┤
│ conv2d_22 (Conv2D)        │ (None, 11, 11,         │         57,456 │
concatenate_2[0]… │
│                           │ 112)                   │                │
│
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────
───────┤
│ conv2d_24 (Conv2D)        │ (None, 11, 11,         │         12,312 │
concatenate_2[0]… │
│                           │ 24)                    │                │
│
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────
───────┤
│ max_pooling2d_6           │ (None, 11, 11,         │              0 │
concatenate_2[0]… │
│ (MaxPooling2D)            │ 512)                   │                │
│
├───────────────────────────┼────────────────────────┼────────────────┼─────────────────
───────┤
│ 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)                 │             │
│
├───────────────────────┼─────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_32 (Conv2D)    │ (None, 11, 11,      │      32,832 │
max_pooling2d_7[… │
│                       │ 64)                 │             │
│
├───────────────────────┼─────────────────────┼─────────────┼──────────────
───────┤
│ concatenate_4         │ (None, 11, 11,      │           0 │ conv2d_27[0]
[0],   │
│ (Concatenate)         │ 512)                │             │ conv2d_29[0]
[0],   │
│                       │                     │             │ conv2d_31[0]
[0],   │
│                       │                     │             │ conv2d_32[0]
[0]    │
├───────────────────────┼─────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_34 (Conv2D)    │ (None, 11, 11,      │      73,872 │
concatenate_4[0]… │
│                       │ 144)                │             │
│
├───────────────────────┼─────────────────────┼─────────────┼──────────────
───────┤
│ conv2d_36 (Conv2D)    │ (None, 11, 11,      │      16,416 │
concatenate_4[0]… │
│                       │ 32)                 │             │
│
├───────────────────────┼─────────────────────┼─────────────┼──────────────
───────┤
│ max_pooling2d_8       │ (None, 11, 11,      │           0 │
concatenate_4[0]… │
│ (MaxPooling2D)        │ 512)                │             │
│
├───────────────────────┼─────────────────────┼─────────────┼──────────────
───────┤
```

```
│ conv2d_33 (Conv2D)  │ (None, 11, 11,    │     57,456 │
concatenate_4[0]… │
│                     │ 112)              │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_35 (Conv2D)  │ (None, 11, 11,    │    373,536 │ conv2d_34[0]
[0]    │
│                     │ 288)              │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_37 (Conv2D)  │ (None, 11, 11,    │     51,264 │ conv2d_36[0]
[0]    │
│                     │ 64)               │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_38 (Conv2D)  │ (None, 11, 11,    │     32,832 │
max_pooling2d_8[… │
│                     │ 64)               │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ concatenate_5       │ (None, 11, 11,    │          0 │ conv2d_33[0]
[0],   │
│ (Concatenate)       │ 528)              │            │ conv2d_35[0]
[0],   │
│                     │                   │            │ conv2d_37[0]
[0],   │
│                     │                   │            │ conv2d_38[0]
[0]    │
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_40 (Conv2D)  │ (None, 11, 11,    │     84,640 │
concatenate_5[0]… │
│                     │ 160)              │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ conv2d_42 (Conv2D)  │ (None, 11, 11,    │     16,928 │
concatenate_5[0]… │
│                     │ 32)               │            │
│
├─────────────────────┼───────────────────┼────────────┼──────────────
──────┤
│ max_pooling2d_9     │ (None, 11, 11,    │          0 │
concatenate_5[0]… │
│ (MaxPooling2D)      │ 528)              │            │
```

```
│
├─────────────────────┼──────────────────┼────────────┼─────────────
───────┤
│ conv2d_39 (Conv2D)  │ (None, 11, 11,   │    135,424 │
concatenate_5[0]… │
│                     │ 256)             │            │
│
├─────────────────────┼──────────────────┼────────────┼─────────────
───────┤
│ 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]
[0]      │
├───────────────────────────┼────────────────────────┼────────────────┼──────────────────
───────┤
│ conv2d_49 (Conv2D)        │ (None, 6, 6, 128)      │        102,528 │ conv2d_48[0]
[0]      │
├───────────────────────────┼────────────────────────┼────────────────┼──────────────────
───────┤
│ conv2d_50 (Conv2D)        │ (None, 6, 6, 128)      │        106,624 │
max_pooling2d_11… │
├───────────────────────────┼────────────────────────┼────────────────┼──────────────────
───────┤
│ concatenate_7             │ (None, 6, 6, 832)      │              0 │ conv2d_45[0]
[0],     │
│ (Concatenate)             │                        │                │ conv2d_47[0]
[0],     │
│                           │                        │                │ conv2d_49[0]
[0],     │
│                           │                        │                │ conv2d_50[0]
[0]      │
├───────────────────────────┼────────────────────────┼────────────────┼──────────────────
───────┤
│ conv2d_52 (Conv2D)        │ (None, 6, 6, 192)      │        159,936 │
concatenate_7[0]… │
├───────────────────────────┼────────────────────────┼────────────────┼──────────────────
───────┤
│ conv2d_54 (Conv2D)        │ (None, 6, 6, 48)       │         39,984 │
concatenate_7[0]… │
├───────────────────────────┼────────────────────────┼────────────────┼──────────────────
───────┤
│ max_pooling2d_12          │ (None, 6, 6, 832)      │              0 │
concatenate_7[0]… │
│ (MaxPooling2D)            │                        │                │
│
├───────────────────────────┼────────────────────────┼────────────────┼──────────────────
───────┤
│ conv2d_51 (Conv2D)        │ (None, 6, 6, 384)      │        319,872 │
```

```
concatenate_7[0]… │
├───────────────────────┼──────────────────────┼───────────────┼──────────────
───────┤
│ conv2d_53 (Conv2D)    │ (None, 6, 6, 384)    │       663,936 │ conv2d_52[0]
[0]     │
├───────────────────────┼──────────────────────┼───────────────┼──────────────
───────┤
│ conv2d_55 (Conv2D)    │ (None, 6, 6, 128)    │       153,728 │ conv2d_54[0]
[0]     │
├───────────────────────┼──────────────────────┼───────────────┼──────────────
───────┤
│ conv2d_56 (Conv2D)    │ (None, 6, 6, 128)    │       106,624 │
max_pooling2d_12… │
├───────────────────────┼──────────────────────┼───────────────┼──────────────
───────┤
│ 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)
```

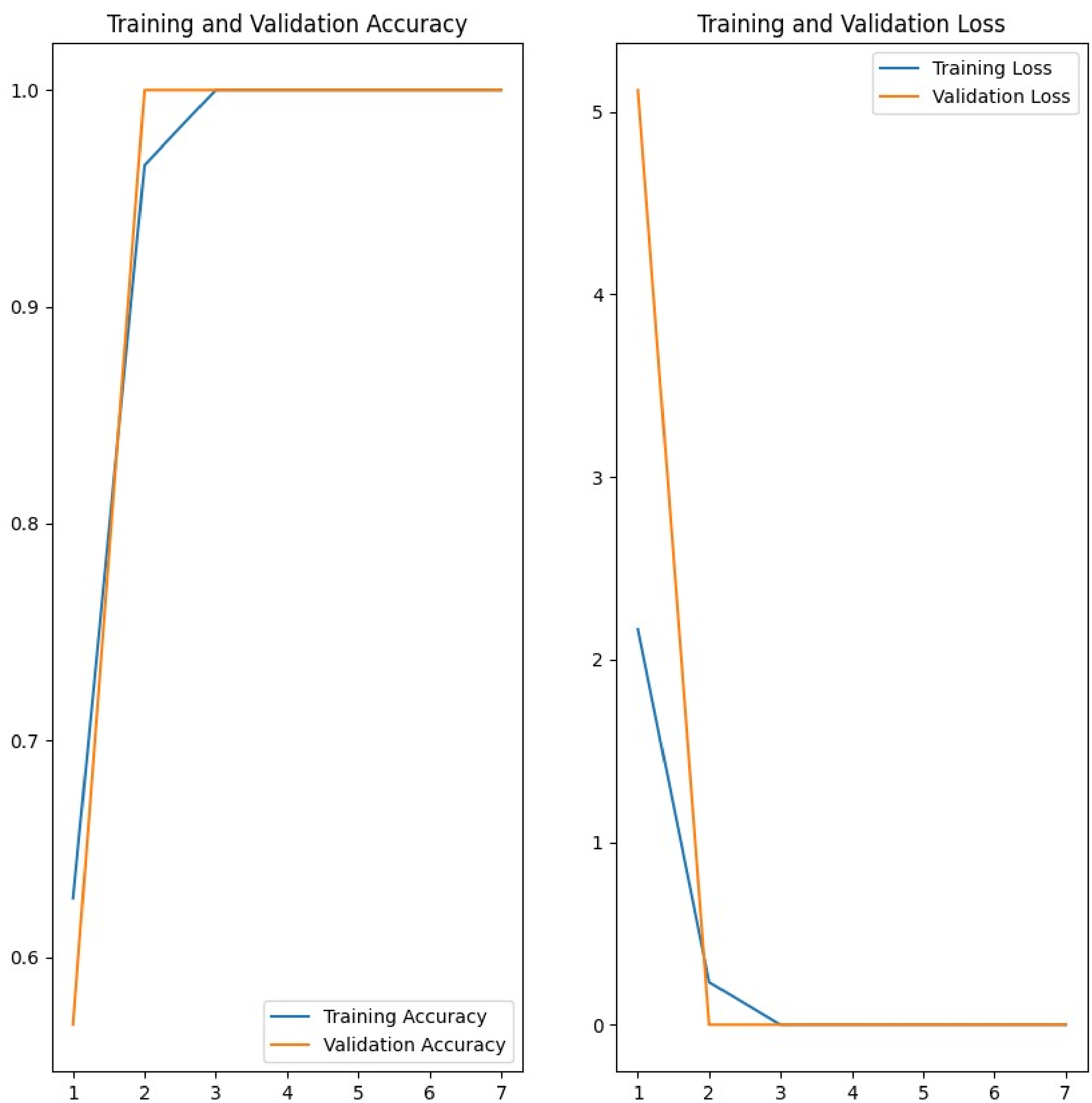
```
 Non-trainable params: 0 (0.00 B)
```

```python
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
#Coimpile dengan optimizer adam
model.compile(
    optimizer=Adam(),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

#buat early stopping
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=5,
                               mode='max')
#fit validation data ke dalam model
history= model.fit(train_ds,
                   epochs=30,
                   validation_data=val_ds,
                   callbacks=[early_stopping])
```

```
Epoch 1/30

45/45 ━━━━━━━━━━━━━━━━━━━━ 58s 847ms/step - accuracy: 0.5315 - loss:
5.8723 - val_accuracy: 0.5688 - val_loss: 5.1193
Epoch 2/30
45/45 ━━━━━━━━━━━━━━━━━━━━ 35s 770ms/step - accuracy: 0.8925 - loss:
0.7784 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 3/30
45/45 ━━━━━━━━━━━━━━━━━━━━ 34s 760ms/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/30
45/45 ━━━━━━━━━━━━━━━━━━━━ 36s 797ms/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 5/30
45/45 ━━━━━━━━━━━━━━━━━━━━ 35s 786ms/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 6/30
45/45 ━━━━━━━━━━━━━━━━━━━━ 36s 796ms/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 7/30
45/45 ━━━━━━━━━━━━━━━━━━━━ 37s 829ms/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
```

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

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

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




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

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

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

```python
from PIL import Image

# Load the trained model
model = load_model(r'D:\semester 5\Mesin Learning\Tugas6_B_11978\
gugelnet.h5')  # Ganti dengan path model Anda
class_names = ['Matang', 'Mentah']

# Function to classify images and save the original image
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        # Load and preprocess the image
        input_image = tf.keras.utils.load_img(image_path,
target_size=(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
result = classify_images(r'D:\semester 5\Mesin Learning\
Tugas6_B_11978\test_data\Mentah\
Immature_Dragon_Original_Data0012.jpg', save_path='mentah2.jpg')
print(result)
```

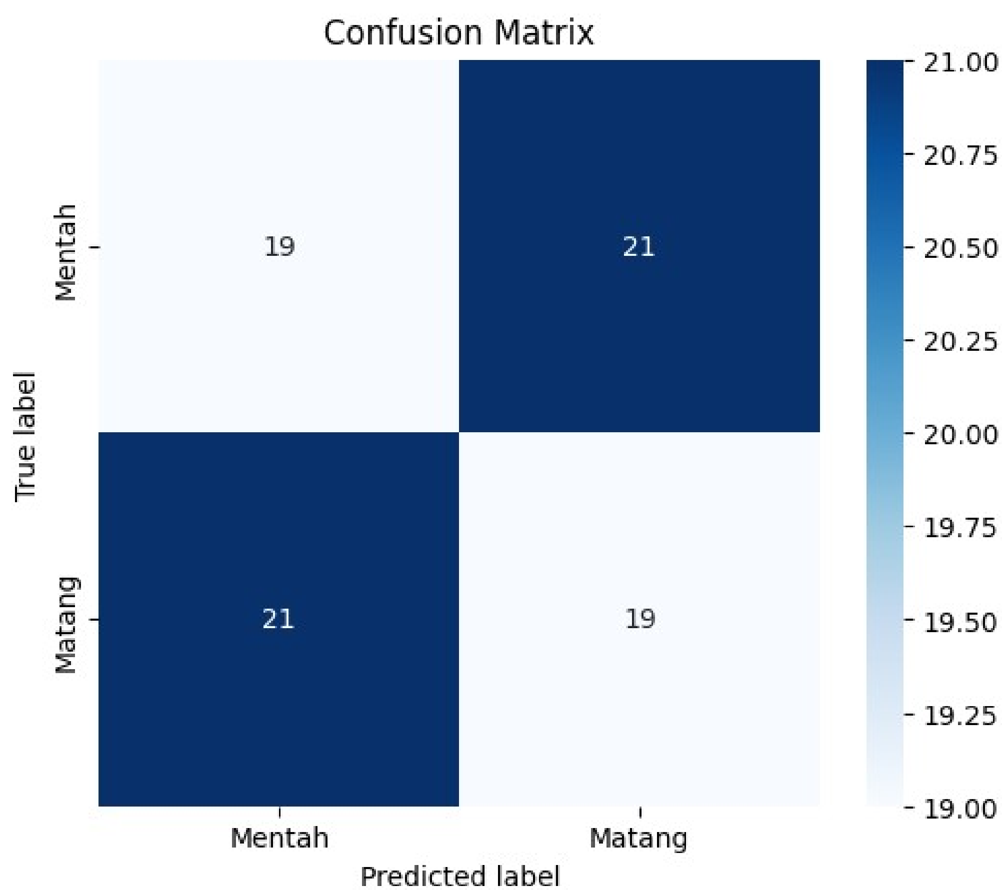
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.

1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
Prediksi: Mentah
Confidence: 73.11%
```

```
Prediksi: Mentah dengan confidence 73.11%. Gambar asli disimpan di
mentah2.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'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
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=["Mentah", "Matang"], yticklabels=["Mentah",
```

```
"Matang"])
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())
```

```
Found 80 files belonging to 2 classes.

3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 210ms/step
```




```
Confusion Matrix:
 [[19 21]
 [21 19]]
Akurasi: 0.475
Presisi: [0.475 0.475]
```

```
Recall: [0.475 0.475]
F1 Score: [0.475 0.475]
```

AlexNet menggunakan TensorFlow dan Keras.

hal yang harus diperhatikan :

1. Data Loading - Data loading meliputi proses membuat fungsi untuk

(1) meload data dari file eksternal dengan parameter lokasi direktori,

(2) ukuran gambar, dan

(3) ukuran batch, di mana

(4) output dari fungsi adalah berupa dataset yang berisi data gambar dan labelnya.

1. Data Checking / Data Visualization - Data visualization meliputi proses membuat fungsi untuk

(1) menampilkan data gambar dengan parameter jumlah gambar yang akan ditampilkan,

(2) ukuran gambar, dan

(3) label gambar, di mana

(4) output dari fungsi adalah berupa visualisasi gambar sesuai dengan parameter input yang ditentukan.

1. Data Preparation - Data preparation meliputi proses (1) normalisasi data gambar dengan benar,

(2) visualisasi sample data hasil normalisasi,

(3) pembagian data menjadi train, validation, dan test, serta

(4) menampilkan ukuran dari masing-masing train, validation and test data

1. Model Architecture - Model architecture meliputi proses

(1) penentuan input,

(2) penentuan layer convolution, pooling, flatten, dan dense dengan tepat,

(3) kompilasi model, dan

(4) menampilkan model summary

1. Model Training - Model training meliputi proses

(1) training model menggunakan sejumlah iterasi/epoch,

(2) menampilkan visualisasi nilai akurasi train dan validation setiap epoch,

(3) menampilkan visualisasi loss train dan validation setiap epoch, serta

(4) menyimpan model yang dihasilkan

1. Model Evaluation - Model evaluation meliputi proses

(1) prediksi untuk setiap data yang ada di test set menggunakan model yang telah disimpan,

(2) menampilkan beberapa contoh hasil prediksi,

(3) mengenerate confusion matrix, dan

(4) menampilkan visualisasi confusion matrix.

1. Model Deployment - Model deployment meliputi proses (1) dump model terbaik,

(2) pembuatan antarmuka aplikasi dengan streamlit,

(3) publikasi ke github, dan

(4) deployment aplikasi di streamlit cloud

Libraries

```
# Import TensorFlow library utama
import tensorflow as tf  # Framework untuk membuat dan melatih model
deep learning

# Import modul Keras untuk membangun model AlexNet
from tensorflow.keras.models import Sequential  # Untuk membuat model
secara berurutan (layer-by-layer)
from tensorflow.keras.layers import Conv2D, MaxPooling2D  # Untuk
layer konvolusi dan pooling
from tensorflow.keras.layers import Flatten, Dense, Dropout  # Untuk
layer fully connected dan regulasi dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator  #
Untuk augmentasi data gambar
from tensorflow.keras.optimizers import Adam  # Optimizer yang sering
digunakan dalam deep learning

# Library tambahan untuk manipulasi data
import numpy as np  # Untuk operasi matematika pada array
import matplotlib.pyplot as plt  # Untuk membuat grafik dan
visualisasi data
import os  # Untuk membaca dan mengelola file/direktori

# Import library untuk membagi dataset
from sklearn.model_selection import train_test_split  # Membagi
dataset menjadi training, validation, dan test sets

# Import library untuk evaluasi performa model
from sklearn.metrics import classification_report, confusion_matrix
# classification_report: Memberikan metrik seperti precision, recall,
F1-score
# confusion_matrix: Untuk menganalisis kesalahan klasifikasi
```

Data Loading

```python
# Import library yang diperlukan
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Fungsi untuk meload dataset
def load_dataset(directory, img_size=(180, 180), batch_size=32):
    """
    Meload data gambar dari direktori eksternal, melakukan augmentasi,
dan menghasilkan dataset dengan labelnya.

    Parameters:
    - directory: Lokasi direktori dataset (string).
    - img_size: Ukuran gambar yang akan diresize (tuple: width,
height), default (224, 224) untuk AlexNet.
    - batch_size: Jumlah data dalam satu batch, default 32.

    Returns:
    - dataset: Objek generator yang berisi data gambar dan labelnya.
    """
    # Membuat instance ImageDataGenerator untuk augmentasi data
    datagen = ImageDataGenerator(
        rescale=1./255,                # Normalisasi nilai piksel (0-
255 menjadi 0-1)
        rotation_range=20,           # Rotasi gambar secara acak
hingga 20 derajat
        width_shift_range=0.2,       # Pergeseran horizontal gambar
hingga 20%
        height_shift_range=0.2,      # Pergeseran vertikal gambar
hingga 20%
        shear_range=0.2,             # Transformasi shear hingga 20%
        zoom_range=0.2,              # Zoom acak hingga 20%
        horizontal_flip=True,        # Membalik gambar secara
horizontal
        fill_mode='nearest'          # Mengisi piksel kosong akibat
augmentasi dengan cara terdekat
    )

    # Menggunakan flow_from_directory untuk membaca gambar dari
direktori
    dataset = datagen.flow_from_directory(
        directory=directory,        # Path direktori dataset
        target_size=img_size,       # Ukuran gambar (disesuaikan
dengan input model)
        batch_size=batch_size,      # Ukuran batch
        class_mode='categorical'    # Label dalam bentuk one-hot
encoding untuk klasifikasi multikelas
    )

    return dataset

# Contoh penggunaan fungsi load_dataset
```

```
# sesuaikan dengan path direktori dataset Anda
train_dataset = load_dataset(directory=r'C:\Users\pejer\OneDrive\
Desktop\UAJY HUB\BISMILLAH S5 UAJY\00 - MATA KULIAH\Pembelejaran Mesin
dan Pembelajaran Mendalam - B\praktek\Tugas6_X_YYYYY\Tugas6_B_11972\
uasML\dataset\dataset[1]\dataset\train_data', img_size=(180, 180),
batch_size=32)
val_dataset = load_dataset(directory=r'C:\Users\pejer\OneDrive\
Desktop\UAJY HUB\BISMILLAH S5 UAJY\00 - MATA KULIAH\Pembelejaran Mesin
dan Pembelajaran Mendalam - B\praktek\Tugas6_X_YYYYY\Tugas6_B_11972\
uasML\dataset\dataset[1]\dataset\validation_data', img_size=(180,
180), batch_size=32)
test_dataset = load_dataset(directory=r'C:\Users\pejer\OneDrive\
Desktop\UAJY HUB\BISMILLAH S5 UAJY\00 - MATA KULIAH\Pembelejaran Mesin
dan Pembelajaran Mendalam - B\praktek\Tugas6_X_YYYYY\Tugas6_B_11972\
uasML\dataset\dataset[1]\dataset\test_data', img_size=(180, 180),
batch_size=32)
```

```
Found 480 images belonging to 3 classes.
Found 60 images belonging to 3 classes.
Found 60 images belonging to 3 classes.
```

Data Visualization

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

def visualize_data(dataset, num_images=9, img_size=(180, 180)):
    """
    Menampilkan visualisasi gambar dari dataset sesuai dengan
parameter yang diberikan.

    Parameters:
    - dataset: Dataset yang berisi gambar dan label.
    - num_images: Jumlah gambar yang akan ditampilkan.
    - img_size: Ukuran gambar yang ditampilkan (tuple: width, height).
    """
    # Ambil batch pertama dari dataset
    images, labels = next(dataset)

    # Menentukan jumlah baris dan kolom untuk plot (misalnya 3x3 untuk
9 gambar)
    num_cols = 3
    num_rows = num_images // num_cols

    # Membuat figure dan axes untuk menampilkan gambar
    fig, axes = plt.subplots(num_rows, num_cols, figsize=(10, 10))

    # Mengatur index gambar yang akan ditampilkan
    for i, ax in enumerate(axes.flat):
        if i < num_images:
```

```
            # Mengambil gambar dan label sesuai dengan index
            img = images[i]
            label = np.argmax(labels[i])  # Mengonversi label one-hot
menjadi label numerik
            label_name = dataset.class_indices
            label_name = {v: k for k, v in label_name.items()}  #
Mengonversi kembali indeks label ke nama kelas
            label = label_name[label]

            # Menampilkan gambar
            ax.imshow(img)
            ax.axis('off')  # Menghilangkan axis
            ax.set_title(f"Label: {label}")  # Menampilkan label
gambar

    plt.tight_layout()
    plt.show()

# Contoh penggunaan fungsi visualisasi dengan dataset pelatihan
visualize_data(train_dataset, num_images=9, img_size=(180, 180))
```


```
# Visualisasi gambar dari dataset validasi
visualize_data(val_dataset, num_images=9, img_size=(180, 180))
```

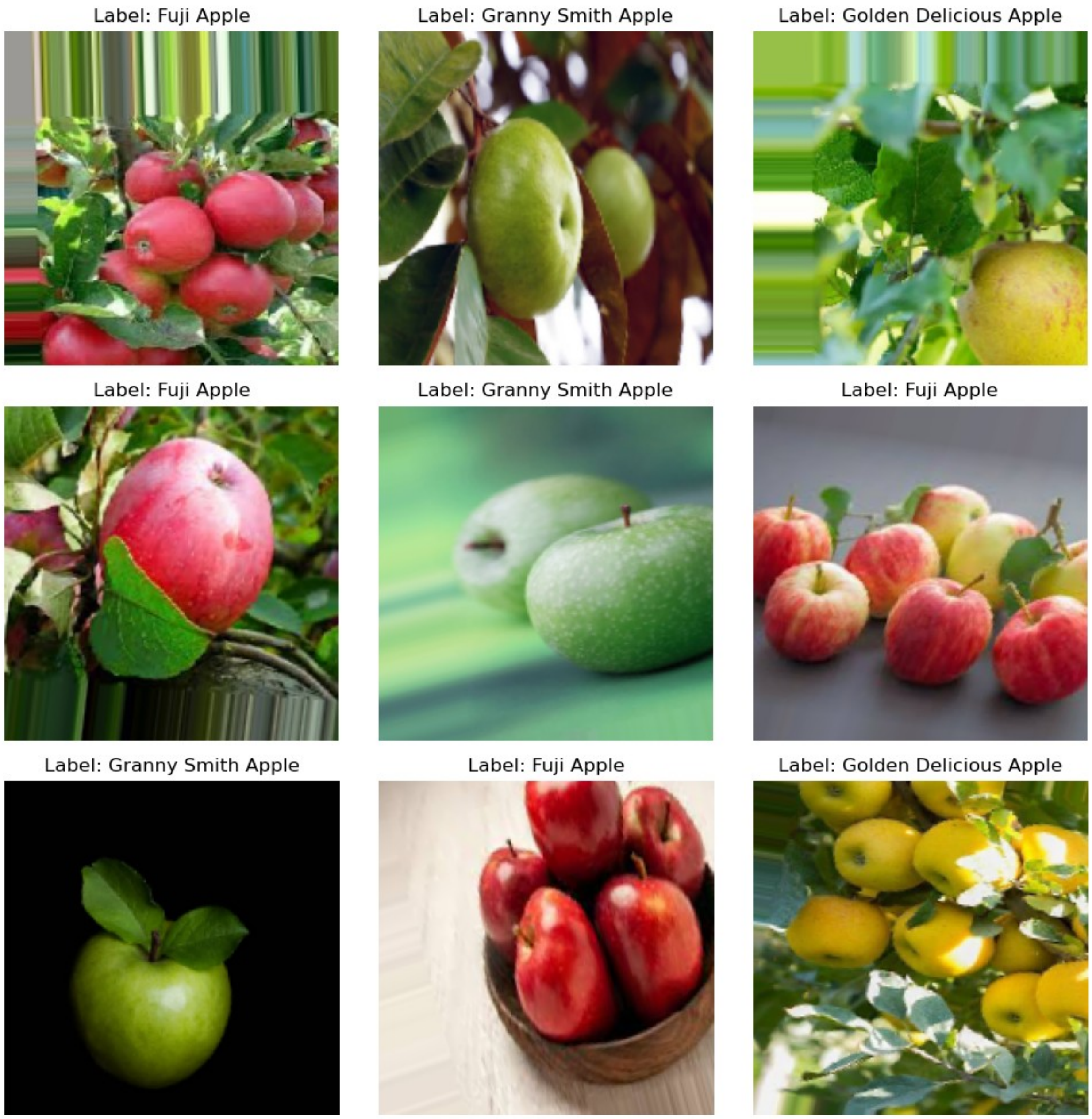

```
# Visualisasi gambar dari dataset pengujian
visualize_data(test_dataset, num_images=9, img_size=(180, 180))
```

Data Preparation

```
# Data Preparation

# 1. Normalisasi Data Gambar
# Sudah dilakukan pada saat loading data menggunakan
ImageDataGenerator dengan rescale=1./255

# 2. Visualisasi Sampel Data Hasil Normalisasi
def visualize_normalized_data(dataset, num_images=9, img_size=(180,
180)):
    """
```

```
    Menampilkan gambar hasil normalisasi dari dataset.

    Parameters:
    - dataset: Dataset yang berisi gambar dan label.
    - num_images: Jumlah gambar yang akan ditampilkan.
    - img_size: Ukuran gambar yang ditampilkan (tuple: width, height).
    """
    # Ambil batch pertama dari dataset
    images, labels = next(dataset)

    # Menentukan jumlah baris dan kolom untuk plot (misalnya 3x3 untuk
9 gambar)
    num_cols = 3
    num_rows = num_images // num_cols

    # Membuat figure dan axes untuk menampilkan gambar
    fig, axes = plt.subplots(num_rows, num_cols, figsize=(10, 10))

    # Mengatur index gambar yang akan ditampilkan
    for i, ax in enumerate(axes.flat):
        if i < num_images:
            # Mengambil gambar dan label sesuai dengan index
            img = images[i]
            label = np.argmax(labels[i])  # Mengonversi label one-hot
menjadi label numerik
            label_name = dataset.class_indices
            label_name = {v: k for k, v in label_name.items()}  #
Mengonversi kembali indeks label ke nama kelas
            label = label_name[label]

            # Menampilkan gambar
            ax.imshow(img)
            ax.axis('off')  # Menghilangkan axis
            ax.set_title(f"Label: {label}")  # Menampilkan label
gambar

    plt.tight_layout()
    plt.show()

# Contoh penggunaan fungsi untuk menampilkan gambar hasil normalisasi
dari dataset pelatihan
visualize_normalized_data(train_dataset, num_images=9, img_size=(180,
180))

# 3. Pembagian Data Menjadi Train, Validation, dan Test
# Kita sudah meload data ke dalam train_dataset, val_dataset, dan
test_dataset melalui fungsi load_dataset

# 4. Menampilkan Ukuran Masing-Masing Dataset
print(f"Ukuran dataset pelatihan: {train_dataset.samples} gambar")
```

```python
print(f"Ukuran dataset validasi: {val_dataset.samples} gambar")
print(f"Ukuran dataset pengujian: {test_dataset.samples} gambar")
```




```
Ukuran dataset pelatihan: 480 gambar
Ukuran dataset validasi: 60 gambar
Ukuran dataset pengujian: 60 gambar
```

Model Architecture - AlexNet

```python
# Cell Block 7: Define AlexNet Architecture
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout, BatchNormalization

# Define the AlexNet architecture
model = Sequential([
    # Layer 1: Convolution + MaxPooling
    Conv2D(96, (11, 11), strides=4, activation='relu',
input_shape=(180, 180, 3)),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3, 3), strides=2),

    # Layer 2: Convolution + MaxPooling
    Conv2D(256, (5, 5), padding='same', activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3, 3), strides=2),

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

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

    # Layer 5: Convolution + MaxPooling
    Conv2D(256, (3, 3), padding='same', activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(3, 3), strides=2),

    # Flatten
    Flatten(),

    # Fully Connected Layers
    Dense(4096, activation='relu'),
    Dropout(0.5),  # Dropout for regularization

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

    Dense(train_dataset.num_classes, activation='softmax')  # Output
layer for classification
])

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

```python
# Print model summary
model.summary()
```

```
Model: "sequential_7"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ conv2d_35 (Conv2D)               │ (None, 43, 43, 96)      │
34,944 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ batch_normalization_5            │ (None, 43, 43, 96)      │
384 │
│ (BatchNormalization)             │                         │
│
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ max_pooling2d_21 (MaxPooling2D)  │ (None, 21, 21, 96)      │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ conv2d_36 (Conv2D)               │ (None, 21, 21, 256)     │
614,656 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ batch_normalization_6            │ (None, 21, 21, 256)     │
1,024 │
│ (BatchNormalization)             │                         │
│
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ max_pooling2d_22 (MaxPooling2D)  │ (None, 10, 10, 256)     │
0 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ conv2d_37 (Conv2D)               │ (None, 10, 10, 384)     │
885,120 │
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
│ batch_normalization_7            │ (None, 10, 10, 384)     │
1,536 │
│ (BatchNormalization)             │                         │
│
├──────────────────────────────────┼─────────────────────────┼──────────
─────┤
```

```
│ conv2d_38 (Conv2D)              │ (None, 10, 10, 384)       │
1,327,488 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ batch_normalization_8           │ (None, 10, 10, 384)       │
1,536 │
│ (BatchNormalization)            │                           │
│
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ conv2d_39 (Conv2D)              │ (None, 10, 10, 256)       │
884,992 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ batch_normalization_9           │ (None, 10, 10, 256)       │
1,024 │
│ (BatchNormalization)            │                           │
│
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ max_pooling2d_23 (MaxPooling2D) │ (None, 4, 4, 256)         │
0 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ flatten_7 (Flatten)             │ (None, 4096)              │
0 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ dense_21 (Dense)                │ (None, 4096)              │
16,781,312 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ dropout_14 (Dropout)            │ (None, 4096)              │
0 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ dense_22 (Dense)                │ (None, 4096)              │
16,781,312 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ dropout_15 (Dropout)            │ (None, 4096)              │
0 │
├─────────────────────────────────┼───────────────────────────┼────────────
────────┤
│ dense_23 (Dense)                │ (None, 3)                 │
12,291 │
└─────────────────────────────────┴───────────────────────────┴────────────
────────┘

 Total params: 37,327,619 (142.39 MB)
```

```
 Trainable params: 37,324,867 (142.38 MB)

 Non-trainable params: 2,752 (10.75 KB)
```

Model Training

```
# Cell Block 8: Train the Model
# Set training parameters
epochs = 15
batch_size = 32

# Train the model
history = model.fit(
    train_dataset,
    validation_data=val_dataset,
    epochs=epochs,
    steps_per_epoch=train_dataset.samples // batch_size,
    validation_steps=val_dataset.samples // batch_size
)
```

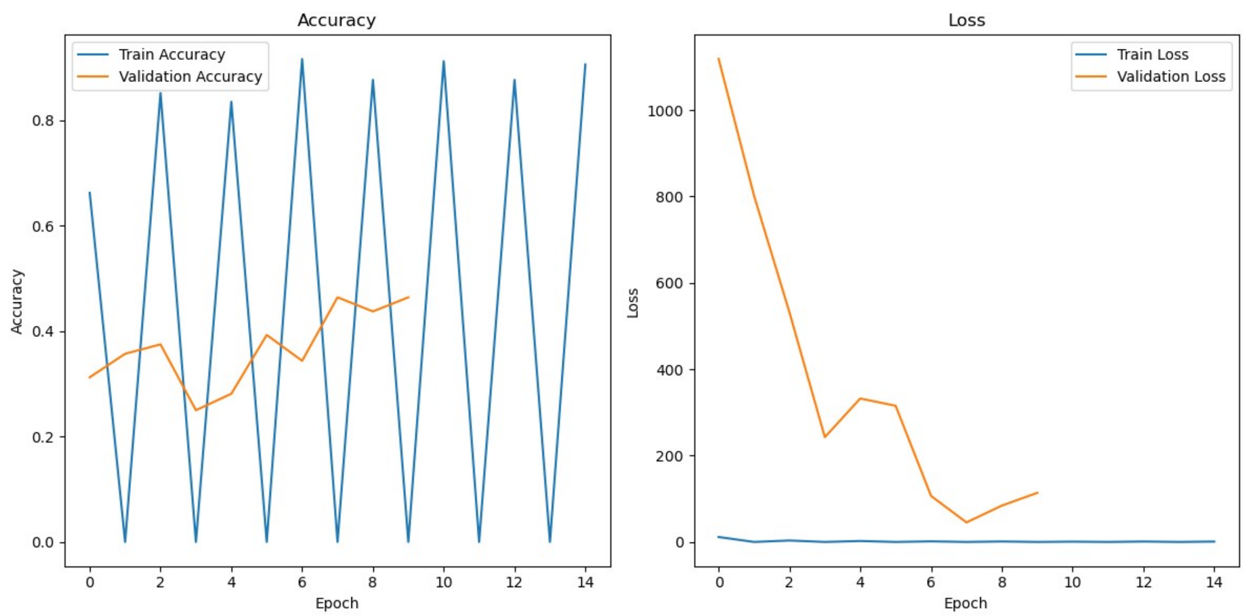
```
Epoch 1/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 26s 1s/step - accuracy: 0.5993 - loss:
14.0676 - val_accuracy: 0.3125 - val_loss: 1118.4553
Epoch 2/15

c:\Users\pejer\anaconda3\Lib\contextlib.py:158: UserWarning: Your
input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs`
batches. You may need to use the `.repeat()` function when building
your dataset.
  self.gen.throw(value)

15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.0000e+00 - loss:
0.0000e+00 - val_accuracy: 0.3571 - val_loss: 801.7838
Epoch 3/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.8799 - loss:
2.8200
Epoch 4/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.0000e+00 - loss:
0.0000e+00 - val_accuracy: 0.3750 - val_loss: 531.7936
Epoch 5/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.8630 - loss:
2.3056 - val_accuracy: 0.2500 - val_loss: 242.6719
Epoch 6/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0000e+00 - loss:
0.0000e+00
Epoch 7/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.9085 - loss:
```

```
1.1537 - val_accuracy: 0.2812 - val_loss: 332.2155
Epoch 8/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.0000e+00 - loss:
0.0000e+00 - val_accuracy: 0.3929 - val_loss: 315.2649
Epoch 9/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 15s 1s/step - accuracy: 0.8779 - loss:
1.1914
Epoch 10/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.0000e+00 - loss:
0.0000e+00 - val_accuracy: 0.3438 - val_loss: 106.5755
Epoch 11/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 0.9183 - loss:
0.5805 - val_accuracy: 0.4643 - val_loss: 45.2076
Epoch 12/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0000e+00 - loss:
0.0000e+00
Epoch 13/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 17s 1s/step - accuracy: 0.8755 - loss:
1.4406 - val_accuracy: 0.4375 - val_loss: 84.1634
Epoch 14/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.0000e+00 - loss:
0.0000e+00 - val_accuracy: 0.4643 - val_loss: 113.7408
Epoch 15/15
15/15 ━━━━━━━━━━━━━━━━━━━━ 17s 1s/step - accuracy: 0.9252 - loss:
0.6099
```

Model Evaluation

```
# Cell Block 9: Evaluate the Model
# Evaluate the model on the test dataset
test_loss, test_accuracy = model.evaluate(test_dataset)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
print(f"Test Loss: {test_loss:.4f}")

c:\Users\pejer\anaconda3\Lib\site-packages\keras\src\trainers\
data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset`
class should call `super().__init__(**kwargs)` in its constructor.
`**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.7826 - loss: 1.3410
Test Accuracy: 78.33%
Test Loss: 1.2506

# Cell Block 10: Visualize Training History
# Visualize training accuracy and loss
plt.figure(figsize=(12, 6))
```

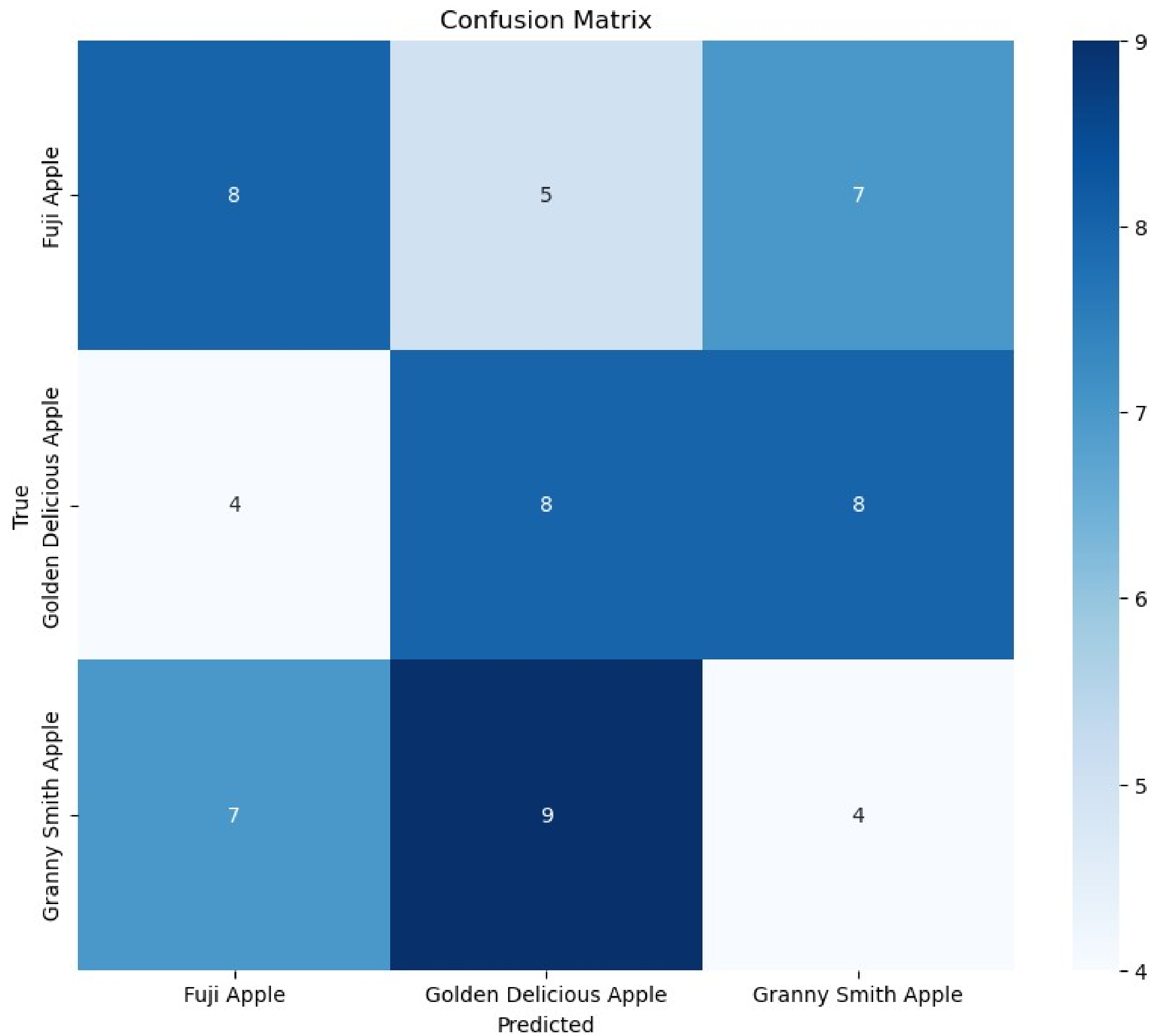
```
# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

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

plt.tight_layout()
plt.show()
```




```
#Cell Block 11: Confusion Matrix and Classification Report
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

# Predict on test dataset
y_true = test_dataset.classes
y_pred = model.predict(test_dataset)
y_pred_classes = np.argmax(y_pred, axis=1)
```

```python
# Classification Report
print("Classification Report:")
print(classification_report(y_true, y_pred_classes,
target_names=list(test_dataset.class_indices.keys())))

# Confusion Matrix
cm = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=test_dataset.class_indices.keys(),
yticklabels=test_dataset.class_indices.keys())
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 6s 3s/step
Classification Report:
                         precision    recall  f1-score   support

            Fuji Apple       0.42      0.40      0.41        20
Golden Delicious Apple       0.36      0.40      0.38        20
    Granny Smith Apple       0.21      0.20      0.21        20

              accuracy                           0.33        60
             macro avg       0.33      0.33      0.33        60
          weighted avg       0.33      0.33      0.33        60
```




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

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

```python
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf

from matplotlib import pyplot as plt
from tensorflow.keras.applications.vgg16 import VGG16,
preprocess_input, decode_predictions # type: ignore
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
Dense, Flatten # type: ignore
from tensorflow.keras.models import Model # type: ignore
from tensorflow.keras.preprocessing import image # type: ignore
from PIL import Image

data_dir = r'D:\Funiversity\Sem5\ML\UAS\dataset'
data = tf.keras.utils.image_dataset_from_directory(data_dir,
image_size=(224, 224), batch_size=16)

print(data.class_names)
class_names = data.class_names
```

```
Found 300 files belonging to 3 classes.
['Fuji Apple', 'Golden Delicious Apple', 'Granny Smith Apple']
```

```python
data_iterator = data.as_numpy_iterator()
print("data_iterator", data_iterator)

batch = data_iterator.next()
print("batch", batch)
```

```
data_iterator
NumpyIterator(iterator=<tensorflow.python.data.ops.iterator_ops.OwnedI
terator object at 0x0000025B597605C0>)
batch (array([[[[115.154015  , 137.29018   ,   1.        ],
         [107.26847   , 131.82143   ,   1.0497249 ],
         [ 98.63123   , 125.22722   ,   1.8830122 ],
         ...,
         [ 91.44033   , 109.57792   ,  58.977802  ],
         [ 91.83767   , 109.934555  ,  57.952503  ],
         [ 95.9979    , 109.131874  ,  52.328037  ]],

        [[126.596985  , 150.56958   ,   2.8944714 ],
         [118.94344   , 143.01302   ,  13.581074  ],
         [118.08705   , 141.80495   ,  30.372252  ],
         ...,
         [ 92.312256  , 109.297554  ,  61.515278  ],
         [ 88.977684  , 105.85413   ,  54.379482  ],
         [ 94.05589   , 107.189865  ,  50.38603   ]],

        [[135.68974   , 160.55582   ,   5.757613  ],
         [133.33293   , 156.93115   ,  31.32228   ],
```

```
         [140.13399   , 161.80363   ,  64.15631   ],
         ...,
         [ 92.63152   , 106.27226   ,  63.283134  ],
         [ 86.815605  ,  99.98798   ,  51.008972  ],
         [ 92.41931   , 105.55328   ,  48.749454  ]],

        ...,

        [[130.24509   ,  96.26248   ,  24.02008   ],
         [117.74771   ,  89.30576   ,  24.309023  ],
         [107.823654  ,  85.68309   ,  28.006813  ],
         ...,
         [ 49.283527  ,  61.32369   ,  25.323689  ],
         [ 50.35769   ,  59.200356  ,  24.991932  ],
         [ 48.888798  ,  55.888798  ,  22.888798  ]],

        [[126.72033   ,  94.663536  ,  21.424072  ],
         [113.51819   ,  87.54424   ,  21.6205    ],
         [104.49601   ,  83.881676  ,  25.684381  ],
         ...,
         [ 50.26028   ,  62.26028   ,  26.260283  ],
         [ 51.18976   ,  59.984375  ,  25.787964  ],
         [ 49.65399   ,  56.65399   ,  23.653992  ]],

        [[122.13001   ,  93.13393   ,  19.270094  ],
         [109.62278   ,  86.20536   ,  19.60045   ],
         [100.2808    ,  81.60091   ,  22.734379  ],
         ...,
         [ 51.595978  ,  63.595978  ,  27.595978  ],
         [ 52.595978  ,  61.390594  ,  27.194183  ],
         [ 51.38216   ,  58.38216   ,  25.382162  ]]],


       [[[113.29241   ,  84.52009   ,  67.95089   ],
         [100.68304   ,  75.73214   ,  52.683037  ],
         [101.4308    ,  83.27679   ,  51.15402   ],
         ...,
         [103.9845    ,  95.0155    ,  30.44629   ],
         [ 92.756744  ,  96.36609   ,  32.36609   ],
         [ 81.91089   ,  87.91089   ,  23.910889  ]],

        [[108.7276    ,  80.680725  ,  60.73205   ],
         [103.28364   ,  78.81043   ,  53.814926  ],
         [ 97.18986   ,  79.44876   ,  44.837128  ],
         ...,
         [ 99.319305  ,  89.93738   ,  26.606966  ],
         [ 90.06707   ,  93.676414  ,  29.676414  ],
         [ 85.60277   ,  91.60277   ,  28.888458  ]],

        [[109.72537   ,  80.92998   ,  56.928497  ],
```

```
        [110.0225    ,  85.00243   ,  56.571552  ],
        [ 97.29559   ,  78.62595   ,  41.136414  ],
        ...,
        [ 94.29925   ,  84.527374  ,  23.760363  ],
        [ 88.29483   ,  91.00459   ,  29.002317  ],
        [ 91.38762   ,  96.302795  ,  35.70231   ]],

       ...,

       [[ 52.968777  ,   0.        ,   0.        ],
        [ 57.37522   ,   0.        ,   0.46881574],
        [ 70.749565  ,   4.7847366 ,   7.865066  ],
        ...,
        [156.84496   ,   2.1612687 ,  21.263796  ],
        [149.43077   ,   0.30130664,  19.102654  ],
        [149.11823   ,   0.45758057,  19.287903  ]],

       [[ 50.61163   ,   0.        ,   0.        ],
        [ 58.352818  ,   0.6563493 ,   2.0559595 ],
        [ 80.77467   ,  11.810368  ,  17.14734   ],
        ...,
        [160.74152   ,   0.62511384,  21.895365  ],
        [154.13394   ,   0.47769177,  20.901794  ],
        [153.4509    ,   0.7254486 ,  20.901794  ]],

       [[ 48.683037  ,   0.        ,   0.        ],
        [ 60.830357  ,   2.3906245 ,   4.781249  ],
        [ 90.50669   ,  18.953121  ,  25.953121  ],
        ...,
        [164.24585   ,   2.2768555 ,  23.122925  ],
        [152.68304   ,   0.        ,  18.        ],
        [152.        ,   0.        ,  18.        ]]],


      [[[254.        , 255.        , 251.        ],
        [254.        , 255.        , 251.        ],
        [255.        , 255.        , 253.        ],
        ...,
        [252.        , 255.        , 255.        ],
        [255.        , 254.        , 255.        ],
        [255.        , 254.        , 255.        ]],

       [[254.        , 255.        , 251.        ],
        [254.        , 255.        , 251.        ],
        [255.        , 255.        , 253.        ],
        ...,
        [252.        , 255.        , 255.        ],
        [255.        , 254.        , 255.        ],
        [255.        , 254.        , 255.        ]],
```

```
        [[254.        , 255.        , 251.        ],
         [254.        , 255.        , 251.        ],
         [255.        , 255.        , 253.        ],
         ...,
         [252.        , 255.        , 255.        ],
         [255.        , 254.        , 255.        ],
         [255.        , 254.        , 255.        ]],

        ...,

        [[255.        , 252.        , 253.        ],
         [255.        , 253.        , 253.        ],
         [254.        , 255.        , 255.        ],
         ...,
         [255.        , 254.        , 255.        ],
         [254.        , 255.        , 255.        ],
         [254.        , 255.        , 255.        ]],

        [[255.        , 252.        , 253.        ],
         [255.        , 253.        , 253.        ],
         [254.        , 255.        , 255.        ],
         ...,
         [255.        , 254.        , 255.        ],
         [254.        , 254.        , 255.        ],
         [252.        , 255.        , 255.        ]],

        [[255.        , 252.        , 253.        ],
         [255.        , 253.        , 253.        ],
         [254.        , 255.        , 255.        ],
         ...,
         [254.        , 255.        , 255.        ],
         [254.        , 254.        , 255.        ],
         [252.        , 255.        , 255.        ]]],


       ...,


       [[[ 71.33482   ,  89.        ,   0.        ],
         [ 79.35519   ,  99.625     ,   5.732422  ],
         [ 82.916855  , 106.81278   ,  10.984097  ],
         ...,
         [ 31.51981   ,  24.541853  ,  21.25      ],
         [ 43.083706  ,  40.230747  ,  34.626118  ],
         [ 60.15402   ,  60.321426  ,  52.321426  ]],

        [[ 73.        ,  89.        ,   7.031249  ],
         [ 79.55831   ,  98.68331   ,  15.464563  ],
         [ 81.6183    , 105.6183    ,  21.271206  ],
         ...,
```

```
         [ 23.092636  ,  19.853237  ,  16.353237  ],
         [ 31.132534  ,  31.26479   ,  25.328127  ],
         [ 31.43527   ,  33.439735  ,  25.941967  ]],

        [[ 73.        ,  89.        ,  24.881699  ],
         [ 76.00921   ,  95.13421   ,  29.242748  ],
         [ 75.31194   ,  99.31194   ,  31.54548   ],
         ...,
         [  2.9006662 ,   2.967352  ,   1.8738813 ],
         [ 10.295198  ,  12.672989  ,   9.95731   ],
         [ 22.022324  ,  26.69643   ,  22.207592  ]],

        ...,

        [[128.57776   , 156.39258   , 124.414795  ],
         [130.43887   , 157.6287    , 127.00276   ],
         [132.85555   , 160.29814   , 132.54999   ],
         ...,
         [191.52315   , 234.32593   , 229.09908   ],
         [194.53981   , 237.0287    , 233.49352   ],
         [199.04443   , 241.53333   , 239.04443   ]],

        [[ 99.05347   , 132.54678   ,  93.05792   ],
         [104.05347   , 137.2329    ,  99.30792   ],
         [111.925125  , 144.72955   , 109.24124   ],
         ...,
         [189.32002   , 234.50668   , 228.19669   ],
         [188.75668   , 233.75668   , 228.75668   ],
         [187.50668   , 232.50668   , 227.50668   ]],

        [[ 69.51517   , 107.013     ,  60.849945  ],
         [ 76.07631   , 113.053764  ,  68.66109   ],
         [ 86.57469   , 122.55268   ,  80.785     ],
         ...,
         [185.64755   , 231.02255   , 224.14755   ],
         [185.897     , 230.897     , 225.897     ],
         [185.16739   , 230.16739   , 225.16739   ]]],


       [[[239.0019    , 213.0019    , 119.7162    ],
         [237.57014   , 212.5711    , 111.28125   ],
         [236.57143   , 210.57143   ,  99.85714   ],
         ...,
         [173.28354   , 143.28354   ,  29.283535  ],
         [172.4263    , 142.4263    ,  28.426296  ],
         [171.99777   , 141.99777   ,  27.997768  ]],

        [[239.85715   , 214.00574   , 116.55899   ],
         [235.57143   , 211.57143   , 104.70855   ],
         [234.56952   , 209.28572   ,  94.12755   ],
```

```
         ...,
         [169.9933    , 139.9933    ,  25.993303  ],
         [168.42183   , 138.42183   ,  24.421831  ],
         [167.9933    , 137.9933    ,  23.993303  ]],

        [[239.8683    , 214.8683    , 113.58259   ],
         [235.154     , 211.58258   , 101.8683    ],
         [233.29688   , 208.29688   ,  89.58259   ],
         ...,
         [167.69829   , 137.69829   ,  25.69829   ],
         [165.98247   , 135.98247   ,  23.982462  ],
         [165.83646   , 135.83646   ,  23.83645   ]],

        ...,

        [[123.13175   , 101.13175   ,  15.133336  ],
         [126.98889   , 105.982544  ,  16.153965  ],
         [129.84921   , 108.84921   ,  15.871426  ],
         ...,
         [223.        , 188.        ,  60.        ],
         [222.42377   , 187.42377   ,  59.423767  ],
         [221.14124   , 186.14124   ,  60.141235  ]],

        [[131.1229    , 106.129555  ,  14.142858  ],
         [134.41527   , 111.41147   ,  17.997149  ],
         [134.9829    , 114.131454  ,  17.428572  ],
         ...,
         [223.0019    , 188.0019    ,  60.0019    ],
         [223.00665   , 188.00665   ,  60.006653  ],
         [223.99335   , 188.99335   ,  60.993347  ]],

        [[135.2873    , 110.2854    ,  17.2854    ],
         [138.99875   , 115.42512   ,  19.428886  ],
         [138.9978    , 118.42575   ,  19.42763   ],
         ...,
         [222.14333   , 187.14333   ,  59.713604  ],
         [221.0022    , 186.0022    ,  58.002197  ],
         [222.8575    , 187.8575    ,  59.56808   ]]],


       [[[ 94.6875    , 112.6875    ,  50.6875    ],
         [ 84.65625   , 102.65625   ,  40.65625   ],
         [ 88.1875    , 106.1875    ,  44.1875    ],
         ...,
         [ 53.71875   ,  64.71875   ,  50.71875   ],
         [ 61.03125   ,  72.03125   ,  58.03125   ],
         [ 79.03125   ,  90.921875  ,  76.921875  ]],

        [[ 89.74651   , 107.74651   ,  45.746513  ],
         [ 88.338905  , 106.338905  ,  44.3389    ],
```

```
          [ 91.33911   , 109.33911   ,  47.33911   ],
          ...,
          [ 52.97991   ,  63.97991   ,  49.97991   ],
          [ 60.29241   ,  71.29241   ,  57.29241   ],
          [ 78.29241   ,  90.18304   ,  76.18304   ]],

         [[ 87.30291   , 105.30291   ,  43.30291   ],
          [ 91.53777   , 109.53777   ,  47.537773  ],
          [ 91.52661   , 109.52661   ,  47.52661   ],
          ...,
          [ 52.15402   ,  63.15402   ,  49.15402   ],
          [ 59.46652   ,  70.466515  ,  56.46652   ],
          [ 75.95763   ,  87.84825   ,  73.84825   ]],

         ...,

         [[ 62.905487  ,  59.905487  ,  43.77603   ],
          [ 62.189697  ,  59.189697  ,  43.06024   ],
          [ 61.517822  ,  58.517822  ,  42.388367  ],
          ...,
          [ 34.014835  ,  48.014835  ,  22.32367   ],
          [ 35.54242   ,  49.83487   ,  22.957523  ],
          [ 40.32367   ,  55.2619    ,  24.896545  ]],

         [[ 57.613342  ,  54.87451   ,  40.39685   ],
          [ 56.79859   ,  54.05976   ,  39.5821    ],
          [ 56.620487  ,  53.881657  ,  39.403996  ],
          ...,
          [ 35.68981   ,  49.68981   ,  23.68981   ],
          [ 37.54516   ,  52.217033  ,  23.850464  ],
          [ 42.497803  ,  57.730408  ,  24.412964  ]],

         [[ 54.        ,  52.        ,  39.        ],
          [ 54.328125  ,  52.328125  ,  39.328125  ],
          [ 55.546875  ,  53.546875  ,  40.546875  ],
          ...,
          [ 36.359375  ,  50.359375  ,  24.359375  ],
          [ 40.015625  ,  54.6875    ,  25.328125  ],
          [ 45.453125  ,  61.34375   ,  25.890625  ]]]],
dtype=float32), array([2, 0, 0, 2, 0, 0, 1, 2, 0, 0, 0, 1, 2, 0, 1,
0], dtype=int32))
```

```
fig, ax = plt.subplots(ncols=8, figsize=(20,20))
for idx, img in enumerate(batch[0][:8]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```


```
data = data.map(lambda x, y: (x / 255.0, y))

print("Tipe data setelah normalisasi: {}".format(data.element_spec))
print("Bentuk data setelah normalisasi: {}".format(data.element_spec))
print("Jumlah data", len(data))
```

```
Tipe data setelah normalisasi: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Bentuk data setelah normalisasi: (TensorSpec(shape=(None, 224, 224,
3), dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Jumlah data 19
```

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

print(train_size)
print(val_size)
print(test_size)

train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size + val_size).take(test_size)
```

```
15
1
1
```

```
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout # type: ignore


def vgg_16():
    model = Sequential()

    model.add(Conv2D(64, (3, 3), padding='same', activation='relu',
input_shape=(227, 227, 3)))
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
```

```
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

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

    model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Flatten())
    model.add(Dense(4096, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(4096, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(3, activation='softmax'))

    return model

model = vgg_16()

model.compile(optimizer='adamax',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
C:\Users\M S I\AppData\Roaming\Python\Python312\site-packages\keras\
src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━
━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━
━━━━━┩
│ conv2d (Conv2D)                  │ (None, 227, 227, 64)    │
1,792 │
├──────────────────────────────────┼─────────────────────────┼──────────
```

```
───────┤
│ conv2d_1 (Conv2D)                │ (None, 227, 227, 64)    │
36,928 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ max_pooling2d (MaxPooling2D)     │ (None, 113, 113, 64)    │
0 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_2 (Conv2D)                │ (None, 113, 113, 128)   │
73,856 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_3 (Conv2D)                │ (None, 113, 113, 128)   │
147,584 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ max_pooling2d_1 (MaxPooling2D)   │ (None, 56, 56, 128)     │
0 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_4 (Conv2D)                │ (None, 56, 56, 256)     │
295,168 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_5 (Conv2D)                │ (None, 56, 56, 256)     │
590,080 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_6 (Conv2D)                │ (None, 56, 56, 256)     │
590,080 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ max_pooling2d_2 (MaxPooling2D)   │ (None, 28, 28, 256)     │
0 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_7 (Conv2D)                │ (None, 28, 28, 512)     │
1,180,160 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_8 (Conv2D)                │ (None, 28, 28, 512)     │
2,359,808 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
│ conv2d_9 (Conv2D)                │ (None, 28, 28, 512)     │
2,359,808 │
├──────────────────────────────────┼─────────────────────────┼────────────
───────┤
```

```
│ max_pooling2d_3 (MaxPooling2D)  │ (None, 14, 14, 512)      │
0 │
├─────────────────────────────────┼──────────────────────────┼────────
───────┤
│ conv2d_10 (Conv2D)              │ (None, 14, 14, 512)      │
2,359,808 │
├─────────────────────────────────┼──────────────────────────┼────────
───────┤
│ conv2d_11 (Conv2D)              │ (None, 14, 14, 512)      │
2,359,808 │
├─────────────────────────────────┼──────────────────────────┼────────
───────┤
│ conv2d_12 (Conv2D)              │ (None, 14, 14, 512)      │
2,359,808 │
├─────────────────────────────────┼──────────────────────────┼────────
───────┤
│ max_pooling2d_4 (MaxPooling2D)  │ (None, 7, 7, 512)        │
0 │
├─────────────────────────────────┼──────────────────────────┼────────
───────┤
│ flatten (Flatten)               │ (None, 25088)            │
0 │
├─────────────────────────────────┼──────────────────────────┼────────
───────┤
│ dense (Dense)                   │ (None, 4096)             │
102,764,544 │
├─────────────────────────────────┼──────────────────────────┼────────
───────┤
│ 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: 134,272,835 (512.21 MB)

 Trainable params: 134,272,835 (512.21 MB)

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

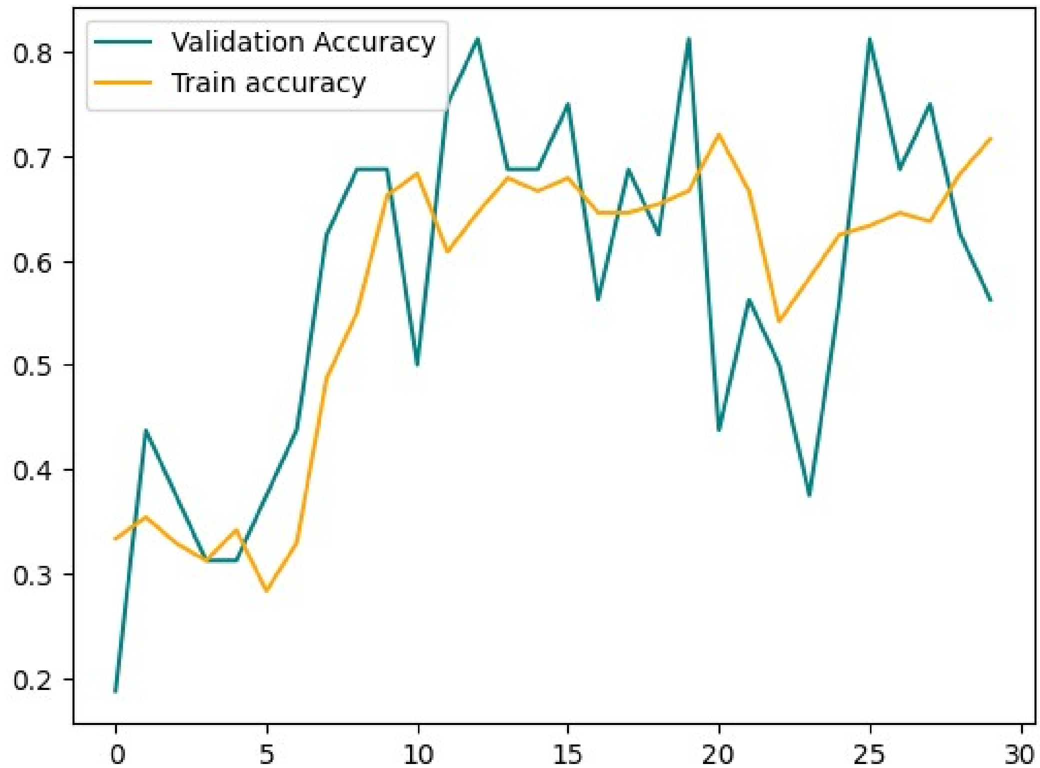
```python
history = model.fit(train, epochs = 30, validation_data=val)
```

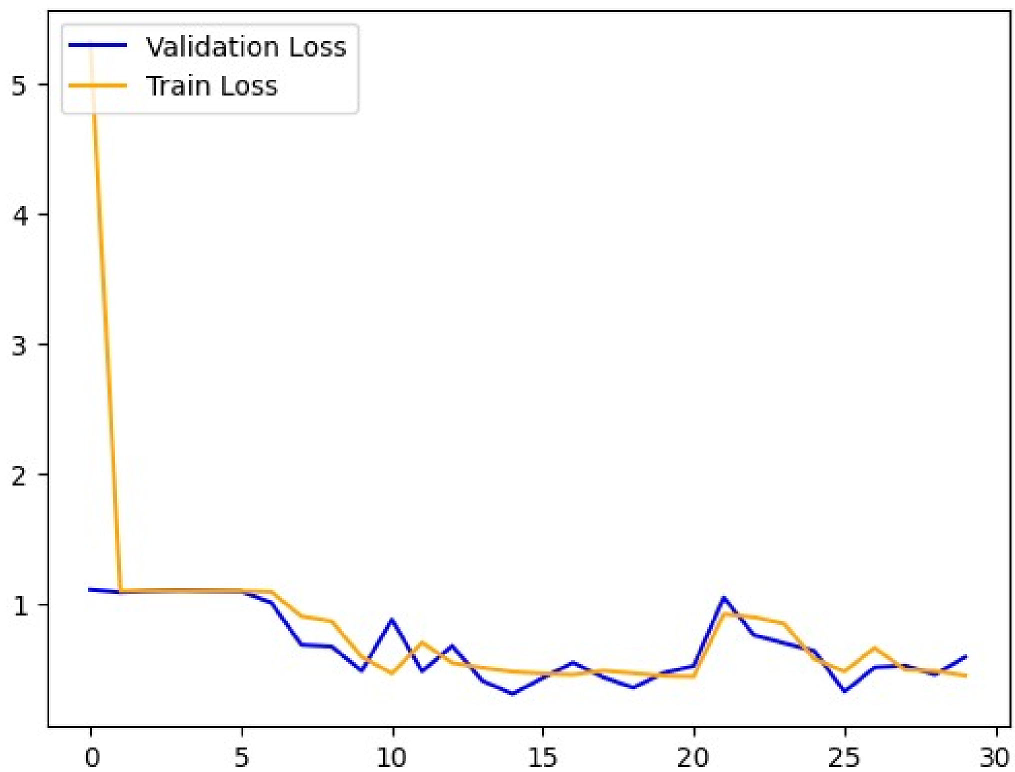
```
Epoch 1/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 72s 5s/step - accuracy: 0.3534 - loss:
10.5359 - val_accuracy: 0.1875 - val_loss: 1.1098
Epoch 2/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 82s 6s/step - accuracy: 0.3737 - loss:
1.0982 - val_accuracy: 0.4375 - val_loss: 1.0891
Epoch 3/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 199s 13s/step - accuracy: 0.3001 - loss:
1.1000 - val_accuracy: 0.3750 - val_loss: 1.0985
Epoch 4/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 145s 10s/step - accuracy: 0.3246 - loss:
1.0986 - val_accuracy: 0.3125 - val_loss: 1.1033
Epoch 5/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 180s 12s/step - accuracy: 0.3077 - loss:
1.0992 - val_accuracy: 0.3125 - val_loss: 1.0994
Epoch 6/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 68s 5s/step - accuracy: 0.2863 - loss:
1.0989 - val_accuracy: 0.3750 - val_loss: 1.0968
Epoch 7/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.3274 - loss:
1.0980 - val_accuracy: 0.4375 - val_loss: 1.0105
Epoch 8/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.4223 - loss:
0.9220 - val_accuracy: 0.6250 - val_loss: 0.6854
Epoch 9/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 205s 14s/step - accuracy: 0.5685 - loss:
0.8604 - val_accuracy: 0.6875 - val_loss: 0.6728
Epoch 10/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 177s 12s/step - accuracy: 0.7309 - loss:
0.5988 - val_accuracy: 0.6875 - val_loss: 0.4848
Epoch 11/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 68s 5s/step - accuracy: 0.6770 - loss:
0.4714 - val_accuracy: 0.5000 - val_loss: 0.8807
Epoch 12/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.6479 - loss:
0.6185 - val_accuracy: 0.7500 - val_loss: 0.4813
Epoch 13/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.6305 - loss:
0.5650 - val_accuracy: 0.8125 - val_loss: 0.6780
Epoch 14/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.6431 - loss:
0.5566 - val_accuracy: 0.6875 - val_loss: 0.4083
Epoch 15/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.7120 - loss:
0.4626 - val_accuracy: 0.6875 - val_loss: 0.3076
Epoch 16/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 71s 5s/step - accuracy: 0.6878 - loss:
0.4425 - val_accuracy: 0.7500 - val_loss: 0.4309
```

```
Epoch 17/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 119s 8s/step - accuracy: 0.6707 - loss:
0.4513 - val_accuracy: 0.5625 - val_loss: 0.5486
Epoch 18/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 148s 9s/step - accuracy: 0.6244 - loss:
0.5048 - val_accuracy: 0.6875 - val_loss: 0.4369
Epoch 19/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 68s 5s/step - accuracy: 0.6640 - loss:
0.4775 - val_accuracy: 0.6250 - val_loss: 0.3554
Epoch 20/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.6749 - loss:
0.4319 - val_accuracy: 0.8125 - val_loss: 0.4692
Epoch 21/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.7371 - loss:
0.4403 - val_accuracy: 0.4375 - val_loss: 0.5226
Epoch 22/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.7196 - loss:
0.7946 - val_accuracy: 0.5625 - val_loss: 1.0499
Epoch 23/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.4786 - loss:
1.0116 - val_accuracy: 0.5000 - val_loss: 0.7604
Epoch 24/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.5727 - loss:
0.7933 - val_accuracy: 0.3750 - val_loss: 0.6984
Epoch 25/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.6325 - loss:
0.5838 - val_accuracy: 0.5625 - val_loss: 0.6389
Epoch 26/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.6339 - loss:
0.4801 - val_accuracy: 0.8125 - val_loss: 0.3246
Epoch 27/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.6446 - loss:
0.6362 - val_accuracy: 0.6875 - val_loss: 0.5120
Epoch 28/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 69s 5s/step - accuracy: 0.6691 - loss:
0.5020 - val_accuracy: 0.7500 - val_loss: 0.5257
Epoch 29/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.7063 - loss:
0.4596 - val_accuracy: 0.6250 - val_loss: 0.4568
Epoch 30/30
15/15 ━━━━━━━━━━━━━━━━━━━━ 70s 5s/step - accuracy: 0.7207 - loss:
0.4720 - val_accuracy: 0.5625 - val_loss: 0.5912
```

```
fig = plt.figure()
plt.plot(history.history['val_accuracy'], color='teal',
label='Validation Accuracy')
plt.plot(history.history['accuracy'], color='orange', label='Train
accuracy')
plt.legend(loc="upper left")
plt.show()
```

```
fig = plt.figure()
plt.plot(history.history['val_loss'], color = 'blue', label =
'Validation Loss')
plt.plot(history.history['loss'], color = 'orange', label = 'Train
Loss')
plt.legend(loc = "upper left")
plt.show()
```






```
model.evaluate(test)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.6875 - loss: 0.3306

[0.33059555292129517, 0.6875]
```

```
model.save(r'D:\Funiversity\Sem5\ML\UAS\Projek UAS PMDPM_B_Pytroch\
BestModel_VGG-16 CNN_PyTorch.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')`.
```