AlexNet

Nama Kelompok: Shogun

Anggota Kelompok: (beserta jobdesknya)

1. Yohani Seprini (210711478) mengumpulkan dan menentukan dataset (untuk train validation test split), mengatasi error pada arsitekture model (AlexNet, GoogleNet, MobileNet, Vgg-16), mengerjakan data preparation (mengubah dataset menjadi iterator numpy, mengambil batch dari iterator,normalisasi data dan menampilkan hasil sebelum dan setelah normalisasi, menghitung jumlah batch dalam dataset, menampilkan visualisasi gambar setelah normalisasi), data augmentasi, implementasi data augmentasi menyimpan akurasi dan loss, mengerjakan model deployment, analisis hasil model dan menentukan model terbaik, dan melakukan deployment pada streamlit
2. Marcella Alicia Ndala (220711907) mengerjakan arsitektur model AlexNet, mengerjakan preprocessing data, menampilkan visualisasi data gambar dari dataset, mengerjakan grafik akurasi dan loss AlexNet
3. Mardika Gidion Omega Limbongan (220712025) mengerjakan arsitektur model GoogleNet, menentukan parameter model alexnet, training model dan memantau proses training, impelementasi early stopping dan callbacks, melakukan penyimpanan model setelah training
4. Aprilius Setio Budi Juja (220712045) mengerjakan arsitektur model MobileNet, impelementasi prediksi untuk dataset uji pada semua model (AlexNet, GoogleNet, MobileNet, VGG-16),menghitung dan menampilkan confusion matrix, menghitung evaluasi metrik model (AlexNet, GoogleNet, MobileNet, VGG-16), visualisasi confusion matrix
5. Jawara Theo Christo (220712066) mengerjakan arsitektur model VGG-16, pengujian model dataset uji untuk semua model (AlexNet, GoogleNet, MobileNet, VGG-16), membantu analisis hasil prediksi, analisis kesalahan prediksi, melakukan perbandingan antar model (AlexNet, GoogleNet, MobileNet, VGG-16)

Data Loading

```
# import library
import os
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
from tensorflow.keras import layers, models
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
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from PIL import Image

# direktori dataset
count = 0
dirs = os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\train')
for dir in dirs:
    files = list(os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\
train/' + dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Garlic Folder has 250 Images
Onion Folder has 250 Images
Red_Onion Folder has 250 Images
Images Folder has 750 Images
```

```
# membaca data dari direktori
base_dir = r'D:\Projek UAS PMDPM SHOGUN\Dataset\train'

validation_split = 0.1

# membaca data dari direktori
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(224, 224),
    batch_size=32,
)

# menampilkan class name
class_names = dataset.class_names
print("Class names:", class_names)
```

```
Found 750 files belonging to 3 classes.
Class names: ['Garlic', 'Onion', 'Red_Onion']
```

```
# train validation test split
total_count = len(list(dataset))
val_count = int(total_count * validation_split)
train_count = total_count - val_count
```

```
test_count = int(len(dataset) * 0.1)

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

train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Data Visualization

```
# menampilkan data gambar dengan paramter jumlah gambar yang
ditampilkan
i = 0
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1) # ukuran gambar
        plt.imshow(images[i].numpy().astype("uint8")) # label gambar
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

Data Preparation

```
# mengubah dataset menjadi iterator numpy
data_iterator = dataset.as_numpy_iterator()
print("data_iterator:", data_iterator)

# mengambil batch berikutnya dari iterator
batch = data_iterator.next()
print("batch:", batch)
```

```
data_iterator:
NumpyIterator(iterator=<tensorflow.python.data.ops.iterator_ops.OwnedI
```

```
terator object at 0x000001BD651B1BD0>)
batch: (array([[[[ 66.96429  ,  39.964287 ,   9.964286 ],
         [ 65.89286  ,  38.892857 ,   8.892858 ],
         [ 64.82143  ,  37.82143  ,   7.821429 ],
         ...,
         [ 82.82143  ,  52.821426 ,  16.821426 ],
         [ 82.10715  ,  52.107147 ,  16.107147 ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 66.01786  ,  39.017857 ,   9.017858 ],
         [ 64.946434 ,  37.946426 ,   7.9464283],
         [ 63.705994 ,  36.705994 ,   6.7059956],
         ...,
         [ 82.04401  ,  52.044006 ,  16.044004 ],
         [ 82.00574  ,  52.00574  ,  16.00574  ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 68.69643  ,  39.875    ,   8.053572 ],
         [ 67.52742  ,  38.705994 ,   6.8845663],
         [ 65.626915 ,  36.805485 ,   4.984056 ],
         ...,
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ]],

        ...,

        [[128.06625  ,  80.066246 ,  32.06625  ],
         [124.27994  ,  76.38708  ,  28.601364 ],
         [122.04784  ,  75.58355  ,  29.40498  ],
         ...,
         [107.780014 ,  60.780014 ,  14.780015 ],
         [111.85843  ,  65.75128  ,  17.072723 ],
         [115.074615 ,  69.074615 ,  20.074615 ]],

        [[120.210464 ,  72.210464 ,  24.210468 ],
         [120.29848  ,  72.40562  ,  24.6199   ],
         [122.3903   ,  75.92601  ,  29.747442 ],
         ...,
         [112.97258  ,  65.97258  ,  19.97258  ],
         [114.89859  ,  68.79144  ,  20.112888 ],
         [116.93239  ,  70.93239  ,  21.932386 ]],

        [[123.92857  ,  75.92857  ,  27.928572 ],
         [121.78571  ,  73.89286  ,  26.107143 ],
         [119.82143  ,  73.35714  ,  27.178572 ],
         ...,
         [115.64285  ,  68.64285  ,  22.642853 ],
         [116.89285  ,  70.785706 ,  22.107147 ],
         [117.       ,  71.       ,  22.       ]]],
```

```
       [[[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        ...,

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
```

```
         [255.       , 255.       , 255.       ]]],


       [[[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 11.390324 ,  11.390324 ,   1.3903232],
         [ 15.337053 ,  11.337053 ,   0.3370536],
         [ 15.337053 ,  11.337053 ,   0.3370536]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 12.011161 ,  12.011161 ,   2.0111609],
         [ 16.       ,  12.       ,   1.       ],
         [ 16.       ,  12.       ,   1.       ]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 13.630205 ,  13.630205 ,   3.6302047],
         [ 16.685268 ,  12.685268 ,   1.6852679],
         [ 16.685268 ,  12.685268 ,   1.6852679]],

        ...,

        [[100.42539  , 106.42539  , 104.42539  ],
         [112.48364  , 118.48364  , 116.48364  ],
         [110.46183  , 116.7029   , 116.62254  ],
         ...,
         [137.68839  , 138.68839  , 133.68839  ],
         [125.148094 , 127.614136 , 120.88112  ],
         [127.43207  , 130.04425  , 123.23816  ]],

        [[100.64094  , 106.64094  , 104.64094  ],
         [ 93.03762  ,  99.03762  ,  97.03762  ],
         [100.624855 , 106.86593  , 104.96678  ],
         ...,
         [136.88385  , 137.72493  , 132.8044   ],
         [135.82593  , 136.82932  , 130.82762  ],
         [135.20984  , 135.       , 129.60492  ]],

        [[ 97.74758  , 103.74758  , 101.74758  ],
         [102.86773  , 108.86773  , 106.86773  ],
         [114.52398  , 120.76505  , 118.84541  ],
         ...,
         [140.7211   , 140.341    , 136.03105  ],
```

```
         [140.0624   , 137.73647  , 132.73647  ],
         [137.39731  , 135.07138  , 130.07138  ]]],


       ...,


       [[[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        ...,

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [254.48987  , 255.       , 250.48987  ],
         [254.28577  , 255.       , 250.28577  ],
         [253.57153  , 254.28577  , 249.57153  ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
```

```
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        ...,
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ]]],


      [[[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       ...,

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],
```

```
        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]]],


       [[[199.38632  , 206.38632  , 222.38632  ],
         [199.09822  , 206.09822  , 222.09822  ],
         [199.32446  , 206.32446  , 222.32446  ],
         ...,
         [212.31535  , 226.98212  , 232.14873  ],
         [213.00351  , 224.19641  , 230.19641  ],
         [210.86845  , 221.86845  , 227.86845  ]],

        [[199.88217  , 206.88217  , 222.88217  ],
         [197.56616  , 204.56616  , 220.56616  ],
         [196.34805  , 203.34805  , 219.34805  ],
         ...,
         [210.44629  , 225.44629  , 230.44629  ],
         [208.56616  , 222.61974  , 228.09294  ],
         [209.98564  , 223.26788  , 229.12677  ]],

        [[198.12196  , 205.12196  , 221.12196  ],
         [198.24124  , 205.24124  , 221.24124  ],
         [196.18478  , 203.18478  , 219.18478  ],
         ...,
         [209.20535  , 222.20535  , 228.20535  ],
         [209.74106  , 222.74106  , 228.74106  ],
         [207.73212  , 220.73212  , 226.73212  ]],

        ...,

        [[166.30693  , 154.30693  , 166.30693  ],
         [160.22354  , 148.04466  , 160.13411  ],
         [153.68944  , 142.7788   , 155.14476  ],
         ...,
         [240.       , 241.       , 236.       ],
         [239.       , 240.       , 235.       ],
         [239.       , 240.       , 235.       ]],

        [[160.50465  , 156.08504  , 167.55824  ],
         [165.44855  , 158.29518  , 172.87187  ],
         [154.28409  , 144.22205  , 159.9094   ],
         ...,
         [240.6607   , 241.6607   , 236.6607   ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]],
```

```
        [[159.2946   , 155.49103  , 165.2946   ],
         [158.72507  , 150.72507  , 165.52512  ],
         [165.58325  , 156.05634  , 171.5653   ],
         ...,
         [239.       , 240.       , 235.       ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]]], dtype=float32),
array([0, 2, 0, 2, 1, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 0, 0, 2, 2, 2, 2,
0,
       0, 2, 1, 1, 1, 1, 0, 1, 2, 0]))

# normalisasi data dengan membagi nilai piksel dengan 255.0
data = dataset.map(lambda x, y: (x/255.0, y))

# tampil tipe data setelah normalisasi
print("Data type after normalization:
{}".format(dataset.element_spec))
# tampil bentuk data setelah normalisasi
print("Data shape after normalization:
{}".format(dataset.element_spec))
# hitung jumlah batch dalam dataset
print("Jumlah images:", len(dataset))
```

```
Data type after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Data shape after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Jumlah images: 24
```

```
# visualisasi gambar setelah normalisasi
plt.figure(figsize=(10, 10))
for images, labels in data.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy())
        plt.title(class_names[labels[i]])
        plt.axis("off")
plt.show()
```


```
# train validation test split
total_count = len(list(dataset))
val_count = int(total_count * validation_split)
train_count = total_count - val_count
test_count = int(len(dataset) * 0.1)

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

```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Model Architecture

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

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

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

# data augmentation
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
])

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

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


```
# AlexNet
modelAlex = Sequential()
modelAlex.add(data_augmentation)
modelAlex.add(Conv2D(96, kernel_size=(11, 11), strides=4,
activation='relu', input_shape=(224, 224, 3)))
modelAlex.add(MaxPooling2D(pool_size=(3, 3), strides=2))
modelAlex.add(Conv2D(256, kernel_size=(5, 5), strides=1,
activation='relu'))
modelAlex.add(MaxPooling2D(pool_size=(3, 3), strides=2))
modelAlex.add(Conv2D(384, kernel_size=(3, 3), strides=1,
activation='relu'))
```

```
modelAlex.add(Conv2D(384, kernel_size=(3, 3), strides=1,
activation='relu'))
modelAlex.add(Conv2D(256, kernel_size=(3, 3), strides=1,
activation='relu'))
modelAlex.add(MaxPooling2D(pool_size=(3, 3), strides=2))
modelAlex.add(Flatten())
modelAlex.add(Dense(4096, activation='relu'))
modelAlex.add(Dropout(0.5))
modelAlex.add(Dense(4096, activation='relu'))
modelAlex.add(Dropout(0.5))
modelAlex.add(Dense(3, activation='softmax'))
```

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

```
# compile model
modelAlex.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

modelAlex.summary()
```

```
Model: "sequential_29"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ sequential_28 (Sequential)           │ (None, 224, 224, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_70 (Conv2D)                   │ (None, 54, 54, 96)          │
34,944 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_42 (MaxPooling2D)      │ (None, 26, 26, 96)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_71 (Conv2D)                   │ (None, 22, 22, 256)         │
```

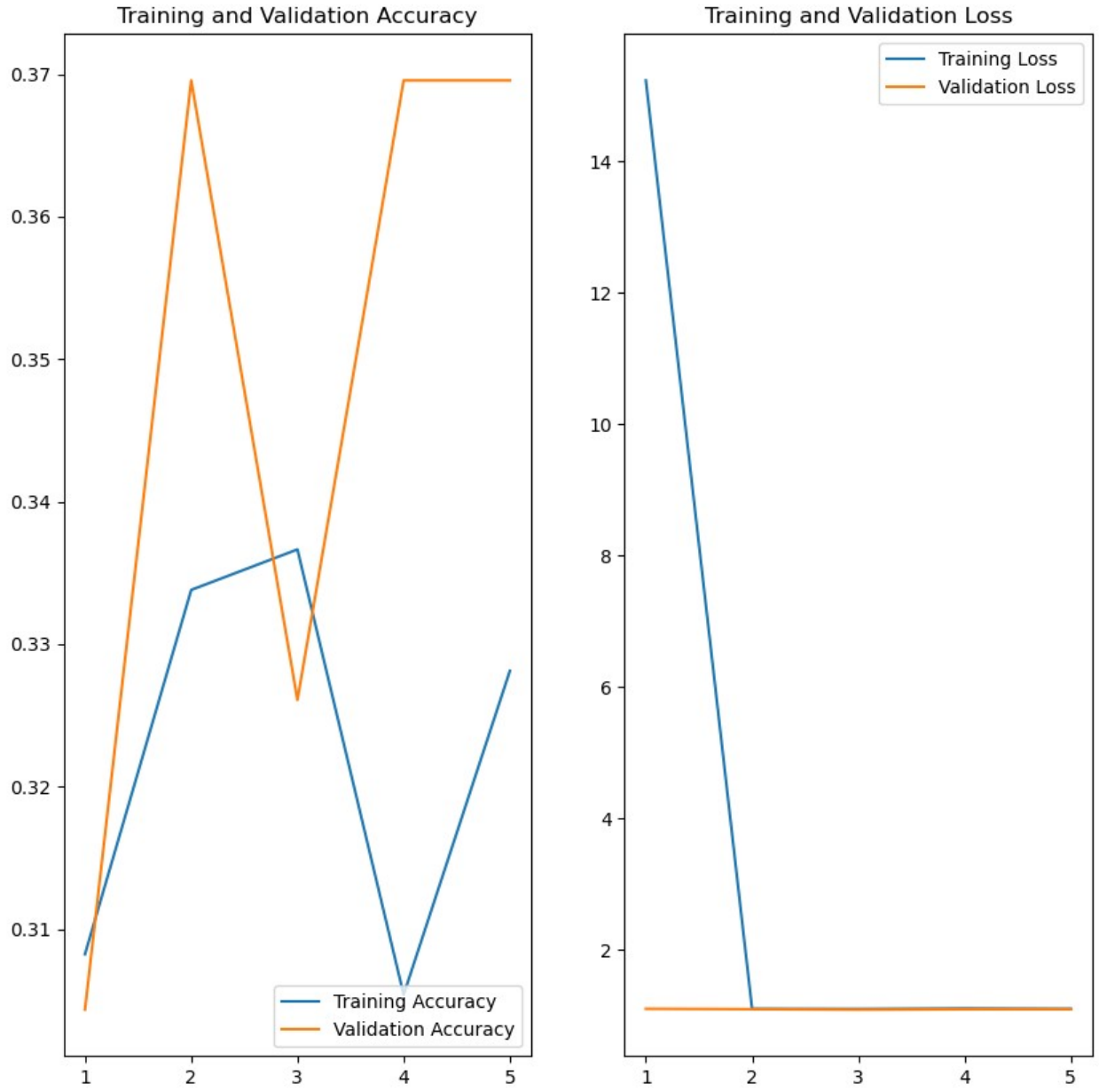
```
614,656 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ max_pooling2d_43 (MaxPooling2D) │ (None, 10, 10, 256)     │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_72 (Conv2D)                  │ (None, 8, 8, 384)       │
885,120 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_73 (Conv2D)                  │ (None, 6, 6, 384)       │
1,327,488 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_74 (Conv2D)                  │ (None, 4, 4, 256)       │
884,992 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ max_pooling2d_44 (MaxPooling2D) │ (None, 1, 1, 256)       │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ flatten_14 (Flatten)                │ (None, 256)             │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_42 (Dense)                    │ (None, 4096)            │
1,052,672 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dropout_28 (Dropout)                │ (None, 4096)            │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_43 (Dense)                    │ (None, 4096)            │
16,781,312 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dropout_29 (Dropout)                │ (None, 4096)            │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_44 (Dense)                    │ (None, 3)               │
12,291 │
└─────────────────────────────────────┴─────────────────────────────┴──────────
──────┘

 Total params: 21,593,475 (82.37 MB)
```

```
 Trainable params: 21,593,475 (82.37 MB)

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

Model Training

```
# training menggunakan iteriasi
history = modelAlex.fit(
    train_ds,
    validation_data=val_ds,
    epochs=100,

callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',
patience=3)]
)

Epoch 1/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 14s 514ms/step - accuracy: 0.3169 - loss:
37.6360 - val_accuracy: 0.3043 - val_loss: 1.1012
Epoch 2/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 11s 489ms/step - accuracy: 0.3272 - loss:
1.1013 - val_accuracy: 0.3696 - val_loss: 1.0955
Epoch 3/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 12s 506ms/step - accuracy: 0.3354 - loss:
1.0986 - val_accuracy: 0.3261 - val_loss: 1.0875
Epoch 4/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 11s 483ms/step - accuracy: 0.3056 - loss:
1.1052 - val_accuracy: 0.3696 - val_loss: 1.0935
Epoch 5/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 11s 493ms/step - accuracy: 0.3417 - loss:
1.0990 - val_accuracy: 0.3696 - val_loss: 1.0953

# menyimpan akurasi dan loss
history_df = pd.DataFrame(history.history)
print(history_df)

   accuracy       loss  val_accuracy  val_loss
0  0.308239  15.225333      0.304348  1.101232
1  0.333807   1.100877      0.369565  1.095541
2  0.336648   1.098526      0.326087  1.087468
3  0.305398   1.105525      0.369565  1.093514
4  0.328125   1.100389      0.369565  1.095305

# visualisasi akurasi dan loss
ephocs_range = range(1, len(history.history['accuracy']) + 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()
```



```
# menyimpan model
modelAlex.save('BestModel_AlexNet_Shogun.h5')
```

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

Model Evaluation

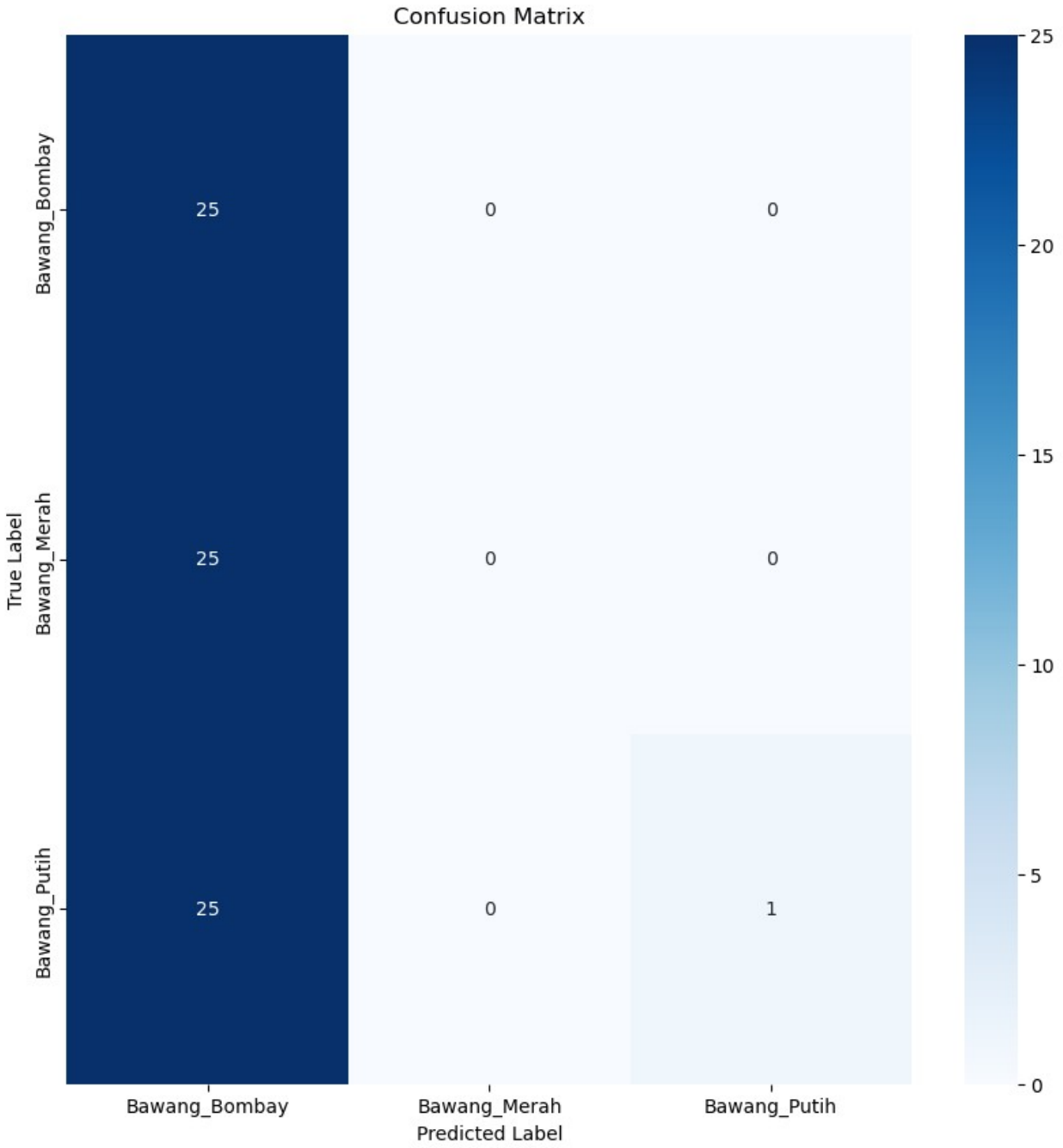
```
# prediksi untuk set data uji
model = load_model('D:\Projek UAS PMDPM SHOGUN\
BestModel_AlexNet_Shogun.h5')
class_names = ['Bawang_Bombay', 'Bawang_Merah', 'Bawang_Putih']

# klasifikasi dataset
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        input_image = tf.keras.utils.load_img(image_path,
target_size=(224, 224))
        input_image_array = tf.keras.utils.img_to_array(input_image) /
255.0
        input_image_exp_dim = tf.expand_dims(input_image_array,
axis=0)

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

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

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

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

result = classify_images(r'D:\Projek UAS PMDPM SHOGUN\Dataset\test\
Red_Onion\Red_Onion_225.jpg', save_path='Red_Onion_AlexNet.jpg')
print(result)
# %
```

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step
Prediction: Bawang_Bombay
Confidence: 33.51%
Prediksi: Bawang_Bombay dengan confidence : 33.51%. Gambar asli
disimpan di Red_Onion_AlexNet.jpg.

# memuat dataset uji
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'D:\Projek UAS PMDPM SHOGUN\dataset\test',
    labels='inferred',
    label_mode='categorical',
    image_size=(224, 224),
    batch_size=32
)

# prediksi dataset uji
y_pred = model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)

# mengambil label sebenarnya
true_labels = []
for images, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())
true_labels = tf.convert_to_tensor(true_labels)

# menghitung confusion matrix
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

# menghitung matrix evaluasi
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)

precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=0)
precision = tf.where(tf.math.is_nan(precision),
tf.zeros_like(precision), precision)

recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=1)
recall = tf.where(tf.math.is_nan(recall), tf.zeros_like(recall),
recall)

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

# menampilkan confusion matrix
plt.figure(figsize=(10, 10))
sns.heatmap(
    conf_mat,
    annot=True,
    fmt='d',
```

```
    cmap='Blues',
    xticklabels=class_names,
    yticklabels=class_names
)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# menampilkan hasil evaluasi
print('Confusion Matrix:\n', conf_mat.numpy())
print('Accuracy:', accuracy.numpy())
print('Precision:', precision.numpy())
print('Recall:', recall.numpy())
print('F1 Score:', f1_score.numpy())
```

```
Found 76 files belonging to 3 classes.
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 152ms/step
```




```
Confusion Matrix:
 [[25  0  0]
 [25  0  0]
 [25  0  1]]
Accuracy: 0.34210526315789475
Precision: [0.33333333 0.         1.        ]
Recall: [1.         0.         0.03846154]
F1 Score: [0.5                nan 0.07407407]
```

GoogleNet

Nama Kelompok: Shogun

Anggota Kelompok: (beserta jobdesknya)

1. Yohani Seprini (210711478) mengumpulkan dan menentukan dataset (untuk train validation test split), mengatasi error pada arsitekture model (AlexNet, GoogleNet, MobileNet, Vgg-16), mengerjakan data preparation (mengubah dataset menjadi iterator numpy, mengambil batch dari iterator,normalisasi data dan menampilkan hasil sebelum dan setelah normalisasi, menghitung jumlah batch dalam dataset, menampilkan visualisasi gambar setelah normalisasi), data augmentasi, implementasi data augmentasi menyimpan akurasi dan loss, mengerjakan model deployment, analisis hasil model dan menentukan model terbaik, dan melakukan deployment pada streamlit
2. Marcella Alicia Ndala (220711907) mengerjakan arsitektur model AlexNet, mengerjakan preprocessing data, menampilkan visualisasi data gambar dari dataset, mengerjakan grafik akurasi dan loss AlexNet
3. Mardika Gidion Omega Limbongan (220712025) mengerjakan arsitektur model GoogleNet, menentukan parameter model alexnet, training model dan memantau proses training, impelementasi early stopping dan callbacks, melakukan penyimpanan model setelah training
4. Aprilius Setio Budi Juja (220712045) mengerjakan arsitektur model MobileNet, impelementasi prediksi untuk dataset uji pada semua model (AlexNet, GoogleNet, MobileNet, VGG-16),menghitung dan menampilkan confusion matrix, menghitung evaluasi metrik model (AlexNet, GoogleNet, MobileNet, VGG-16), visualisasi confusion matrix
5. Jawara Theo Christo (220712066) mengerjakan arsitektur model VGG-16, pengujian model dataset uji untuk semua model (AlexNet, GoogleNet, MobileNet, VGG-16), membantu analisis hasil prediksi, analisis kesalahan prediksi, melakukan perbandingan antar model (AlexNet, GoogleNet, MobileNet, VGG-16)

Data Loading

```
# import library
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras._tf_keras.keras.backend as K
import cv2
import os
```

```
import keras
import seaborn as sns

from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model
from keras._tf_keras.keras.models import  Model, load_model
from keras._tf_keras.keras.layers import Input, Dense, Conv2D,
Flatten, MaxPooling2D, AvgPool2D
from keras._tf_keras.keras.layers import Concatenate, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from PIL import Image

# direktori dataset
count = 0
dirs = os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\train')
for dir in dirs:
    files = list(os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\
train/' + dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Garlic Folder has 250 Images
Onion Folder has 250 Images
Red_Onion Folder has 250 Images
Images Folder has 750 Images
```

```
# direktori dataset
base_dir = r'D:\Projek UAS PMDPM SHOGUN\Dataset\train'
validation_split = 0.1

# membuat dataset
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(224, 224),
    batch_size=32,
    shuffle=True,
)

# class names
class_names = dataset.class_names
print("Class names:", class_names)
```

```
Found 750 files belonging to 3 classes.
Class names: ['Garlic', 'Onion', 'Red_Onion']
```

```
# train validation test split
total_count = len(list(dataset))
val_count = int(total_count * validation_split)
```

```
train_count = total_count - val_count
test_count = int(len(dataset) * 0.1)

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

train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Data Visualization

```
# menampilkan data gambar dengan paramter jumlah gambar yang ditampilkan
i = 0
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1) # ukuran gambar
        plt.imshow(images[i].numpy().astype("uint8")) # label gambar
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

Data Preparation

```
# ubah dataset menjadi iterator numpy
data_iterator = dataset.as_numpy_iterator()
print("data_iterator:", data_iterator)

# ambil batch berikutnya dari iterator
batch = data_iterator.next()
print("batch:", batch)
```

```
data_iterator:
NumpyIterator(iterator=<tensorflow.python.data.ops.iterator_ops.OwnedI
```

```
terator object at 0x000001C29E597DD0>)
batch: (array([[[[ 66.96429  ,  39.964287 ,   9.964286 ],
         [ 65.89286  ,  38.892857 ,   8.892858 ],
         [ 64.82143  ,  37.82143  ,   7.821429 ],
         ...,
         [ 82.82143  ,  52.821426 ,  16.821426 ],
         [ 82.10715  ,  52.107147 ,  16.107147 ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 66.01786  ,  39.017857 ,   9.017858 ],
         [ 64.946434 ,  37.946426 ,   7.9464283],
         [ 63.705994 ,  36.705994 ,   6.7059956],
         ...,
         [ 82.04401  ,  52.044006 ,  16.044004 ],
         [ 82.00574  ,  52.00574  ,  16.00574  ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 68.69643  ,  39.875    ,   8.053572 ],
         [ 67.52742  ,  38.705994 ,   6.8845663],
         [ 65.626915 ,  36.805485 ,   4.984056 ],
         ...,
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ]],

        ...,

        [[128.06625  ,  80.066246 ,  32.06625  ],
         [124.27994  ,  76.38708  ,  28.601364 ],
         [122.04784  ,  75.58355  ,  29.40498  ],
         ...,
         [107.780014 ,  60.780014 ,  14.780015 ],
         [111.85843  ,  65.75128  ,  17.072723 ],
         [115.074615 ,  69.074615 ,  20.074615 ]],

        [[120.210464 ,  72.210464 ,  24.210468 ],
         [120.29848  ,  72.40562  ,  24.6199   ],
         [122.3903   ,  75.92601  ,  29.747442 ],
         ...,
         [112.97258  ,  65.97258  ,  19.97258  ],
         [114.89859  ,  68.79144  ,  20.112888 ],
         [116.93239  ,  70.93239  ,  21.932386 ]],

        [[123.92857  ,  75.92857  ,  27.928572 ],
         [121.78571  ,  73.89286  ,  26.107143 ],
         [119.82143  ,  73.35714  ,  27.178572 ],
         ...,
         [115.64285  ,  68.64285  ,  22.642853 ],
         [116.89285  ,  70.785706 ,  22.107147 ],
         [117.       ,  71.       ,  22.       ]]],
```

```
       [[[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        ...,

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
```

```
         [255.       , 255.       , 255.       ]]],


       [[[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 11.390324 ,  11.390324 ,   1.3903232],
         [ 15.337053 ,  11.337053 ,   0.3370536],
         [ 15.337053 ,  11.337053 ,   0.3370536]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 12.011161 ,  12.011161 ,   2.0111609],
         [ 16.       ,  12.       ,   1.       ],
         [ 16.       ,  12.       ,   1.       ]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 13.630205 ,  13.630205 ,   3.6302047],
         [ 16.685268 ,  12.685268 ,   1.6852679],
         [ 16.685268 ,  12.685268 ,   1.6852679]],

        ...,

        [[100.42539  , 106.42539  , 104.42539  ],
         [112.48364  , 118.48364  , 116.48364  ],
         [110.46183  , 116.7029   , 116.62254  ],
         ...,
         [137.68839  , 138.68839  , 133.68839  ],
         [125.148094 , 127.614136 , 120.88112  ],
         [127.43207  , 130.04425  , 123.23816  ]],

        [[100.64094  , 106.64094  , 104.64094  ],
         [ 93.03762  ,  99.03762  ,  97.03762  ],
         [100.624855 , 106.86593  , 104.96678  ],
         ...,
         [136.88385  , 137.72493  , 132.8044   ],
         [135.82593  , 136.82932  , 130.82762  ],
         [135.20984  , 135.       , 129.60492  ]],

        [[ 97.74758  , 103.74758  , 101.74758  ],
         [102.86773  , 108.86773  , 106.86773  ],
         [114.52398  , 120.76505  , 118.84541  ],
         ...,
         [140.7211   , 140.341    , 136.03105  ],
```

```
         [140.0624   , 137.73647  , 132.73647  ],
         [137.39731  , 135.07138  , 130.07138  ]]],


       ...,


       [[[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        ...,

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [254.48987  , 255.       , 250.48987  ],
         [254.28577  , 255.       , 250.28577  ],
         [253.57153  , 254.28577  , 249.57153  ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
```

```
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        ...,
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ]]],


      [[[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       ...,

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],
```

```
        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]]],


       [[[199.38632  , 206.38632  , 222.38632  ],
         [199.09822  , 206.09822  , 222.09822  ],
         [199.32446  , 206.32446  , 222.32446  ],
         ...,
         [212.31535  , 226.98212  , 232.14873  ],
         [213.00351  , 224.19641  , 230.19641  ],
         [210.86845  , 221.86845  , 227.86845  ]],

        [[199.88217  , 206.88217  , 222.88217  ],
         [197.56616  , 204.56616  , 220.56616  ],
         [196.34805  , 203.34805  , 219.34805  ],
         ...,
         [210.44629  , 225.44629  , 230.44629  ],
         [208.56616  , 222.61974  , 228.09294  ],
         [209.98564  , 223.26788  , 229.12677  ]],

        [[198.12196  , 205.12196  , 221.12196  ],
         [198.24124  , 205.24124  , 221.24124  ],
         [196.18478  , 203.18478  , 219.18478  ],
         ...,
         [209.20535  , 222.20535  , 228.20535  ],
         [209.74106  , 222.74106  , 228.74106  ],
         [207.73212  , 220.73212  , 226.73212  ]],

        ...,

        [[166.30693  , 154.30693  , 166.30693  ],
         [160.22354  , 148.04466  , 160.13411  ],
         [153.68944  , 142.7788   , 155.14476  ],
         ...,
         [240.       , 241.       , 236.       ],
         [239.       , 240.       , 235.       ],
         [239.       , 240.       , 235.       ]],

        [[160.50465  , 156.08504  , 167.55824  ],
         [165.44855  , 158.29518  , 172.87187  ],
         [154.28409  , 144.22205  , 159.9094   ],
         ...,
         [240.6607   , 241.6607   , 236.6607   ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]],
```

```
        [[159.2946   , 155.49103  , 165.2946   ],
         [158.72507  , 150.72507  , 165.52512  ],
         [165.58325  , 156.05634  , 171.5653   ],
         ...,
         [239.       , 240.       , 235.       ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]]], dtype=float32),
array([0, 2, 0, 2, 1, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 0, 0, 2, 2, 2, 2,
0,
       0, 2, 1, 1, 1, 1, 0, 1, 2, 0]))
```

```
# normalisasi data dengan membagi nilai piksel dengan 255.0
data = dataset.map(lambda x, y: (x/255.0, y))

# tampil tipe data setelah normalisasi
print("Data type after normalization:
{}".format(dataset.element_spec))
# tampil bentuk data setelah normalisasi
print("Data shape after normalization:
{}".format(dataset.element_spec))
# hitung jumlah batch dalam dataset
print("Jumlah images:", len(dataset))
```

```
Data type after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Data shape after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Jumlah images: 24
```

```
# visualisasi setelah normalisasi
plt.figure(figsize=(10, 10))
for images, labels in data.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy())
        plt.title(class_names[labels[i]])
        plt.axis("off")
plt.show()
```


```
# train validation test split
total_count = len(list(dataset))
val_count = int(total_count * validation_split)
train_count = total_count - val_count
test_count = int(len(dataset) * 0.1)

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

```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Model Architecture

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

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

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

# pre-trained (InceptionV1)
base_model = tf.keras.applications.InceptionV3(
    weights='imagenet',
    include_top=False,
    input_shape=(224, 224, 3)
)

base_model.trainable = True # False = Freeze the base model

for layer in base_model.layers[:-20]:
    layer.trainable = False

# augmentation data
data_augmentation = Sequential([
    layers.RandomFlip("horizontal_and_vertical", input_shape=(224,
224, 3)),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
    layers.RandomTranslation(height_factor=0.1, width_factor=0.1),
    layers.RandomContrast(0.2),
    layers.RandomBrightness(0.1)
])

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

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
```

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

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

```python
# GoogleNet
def googlenet(input_shape, n_classes):
    modelGoogleNet = Sequential()
    modelGoogleNet.add(Conv2D(32, (3, 3), activation='relu',
input_shape=input_shape))
    modelGoogleNet.add(MaxPooling2D(pool_size=(2, 2)))
    modelGoogleNet.add(Conv2D(64, (3, 3), activation='relu'))
    modelGoogleNet.add(MaxPooling2D(pool_size=(2, 2)))
    modelGoogleNet.add(Conv2D(128, (3, 3), activation='relu'))
    modelGoogleNet.add(MaxPooling2D(pool_size=(2, 2)))
    modelGoogleNet.add(Flatten())
    modelGoogleNet.add(Dense(128, activation='relu'))
    modelGoogleNet.add(Dropout(0.5))
    modelGoogleNet.add(Dense(n_classes, activation='softmax'))
    return modelGoogleNet

input_shape= (224, 224, 3)
n_classes= 3
K.clear_session()
modelGoogleNet = googlenet(input_shape, n_classes)

modelGoogleNet.compile(
    optimizer=Adam(learning_rate=1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

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

```python
modelGoogleNet.summary()
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━
┩
│ conv2d (Conv2D)                      │ (None, 222, 222, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
│ max_pooling2d (MaxPooling2D)         │ (None, 111, 111, 32)        │
```

```
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_1 (Conv2D)                   │ (None, 109, 109, 64)        │
18,496 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ max_pooling2d_1 (MaxPooling2D)      │ (None, 54, 54, 64)          │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ conv2d_2 (Conv2D)                   │ (None, 52, 52, 128)         │
73,856 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ max_pooling2d_2 (MaxPooling2D)      │ (None, 26, 26, 128)         │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ flatten (Flatten)                   │ (None, 86528)               │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense (Dense)                       │ (None, 128)                 │
11,075,712 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dropout (Dropout)                   │ (None, 128)                 │
0 │
├─────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
│ dense_1 (Dense)                     │ (None, 3)                   │
387 │
└─────────────────────────────────────┴─────────────────────────────┴──────────
──────┘

 Total params: 11,169,347 (42.61 MB)

 Trainable params: 11,169,347 (42.61 MB)

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

Model Training

```python
# training menggunakan iteriasi
early_stopping = EarlyStopping(
    monitor='val_accuracy',
    patience=5,
    mode='max',
)
```

```
history = modelGoogleNet.fit(
    train_ds,
    validation_data=val_ds,
    epochs=100,
    callbacks=[early_stopping]
)
```

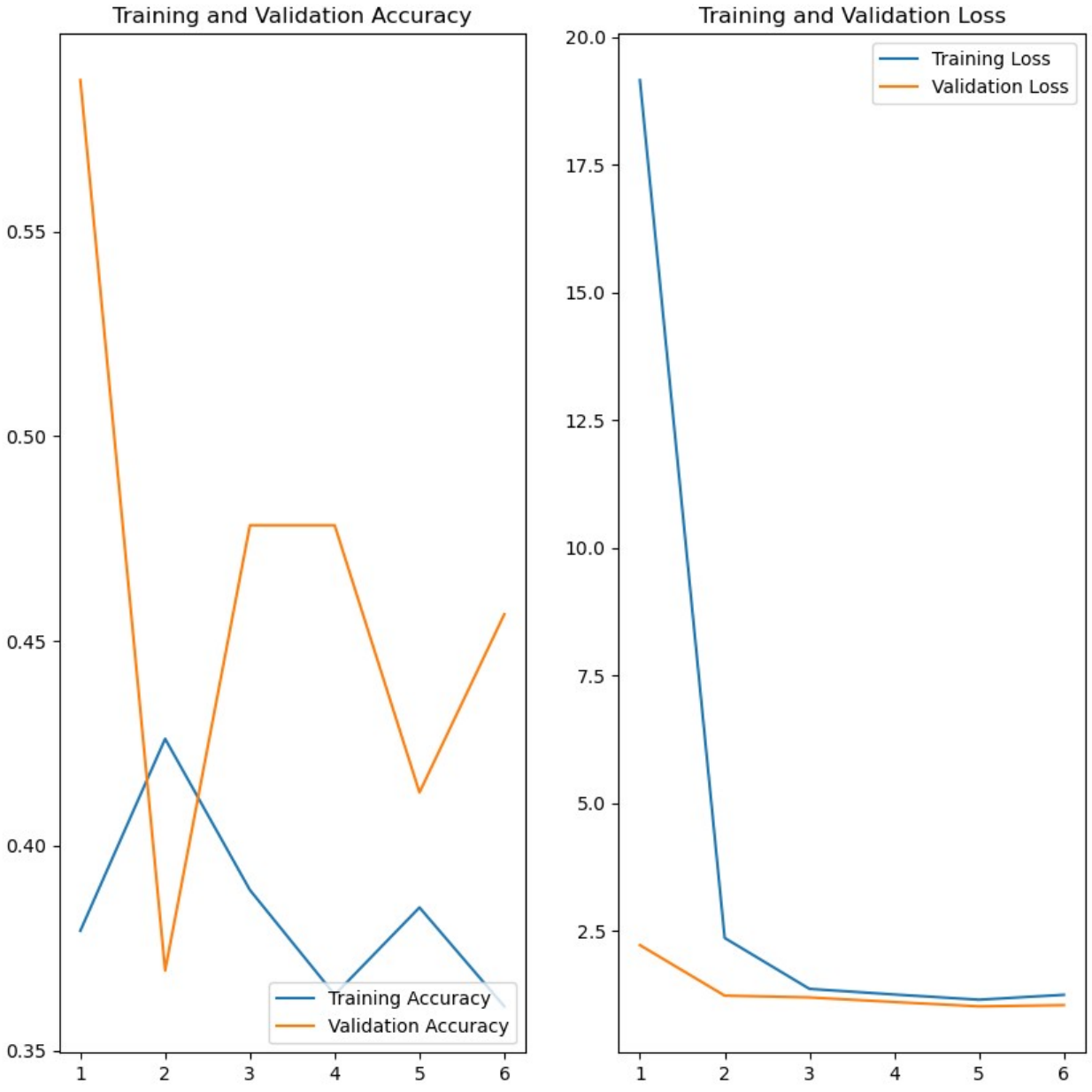
```
Epoch 1/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 17s 679ms/step - accuracy: 0.3481 - loss:
23.0931 - val_accuracy: 0.5870 - val_loss: 2.2143
Epoch 2/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 15s 650ms/step - accuracy: 0.4541 - loss:
3.3647 - val_accuracy: 0.3696 - val_loss: 1.2240
Epoch 3/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 15s 673ms/step - accuracy: 0.4015 - loss:
1.5001 - val_accuracy: 0.4783 - val_loss: 1.1889
Epoch 4/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 17s 739ms/step - accuracy: 0.3684 - loss:
1.1923 - val_accuracy: 0.4783 - val_loss: 1.0991
Epoch 5/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 17s 765ms/step - accuracy: 0.3774 - loss:
1.1487 - val_accuracy: 0.4130 - val_loss: 1.0137
Epoch 6/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 16s 717ms/step - accuracy: 0.3489 - loss:
1.2059 - val_accuracy: 0.4565 - val_loss: 1.0375
```

```
# menyimpan akurasi dan loss
history_df = pd.DataFrame(history.history)
print(history_df)
```

```
   accuracy       loss  val_accuracy  val_loss
0  0.379261  19.160936      0.586957  2.214314
1  0.426136   2.353983      0.369565  1.223970
2  0.389205   1.356427      0.478261  1.188906
3  0.363636   1.246505      0.478261  1.099051
4  0.384943   1.144151      0.413043  1.013681
5  0.360795   1.240396      0.456522  1.037490
```

```
# visualisasi akurasi dan loss
ephocs_range = range(1, len(history.history['accuracy']) + 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()
```




```
# save model
modelGoogleNet.save('BestModel_GoogleNet_Shogun.h5')
```

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

Model Evaluation

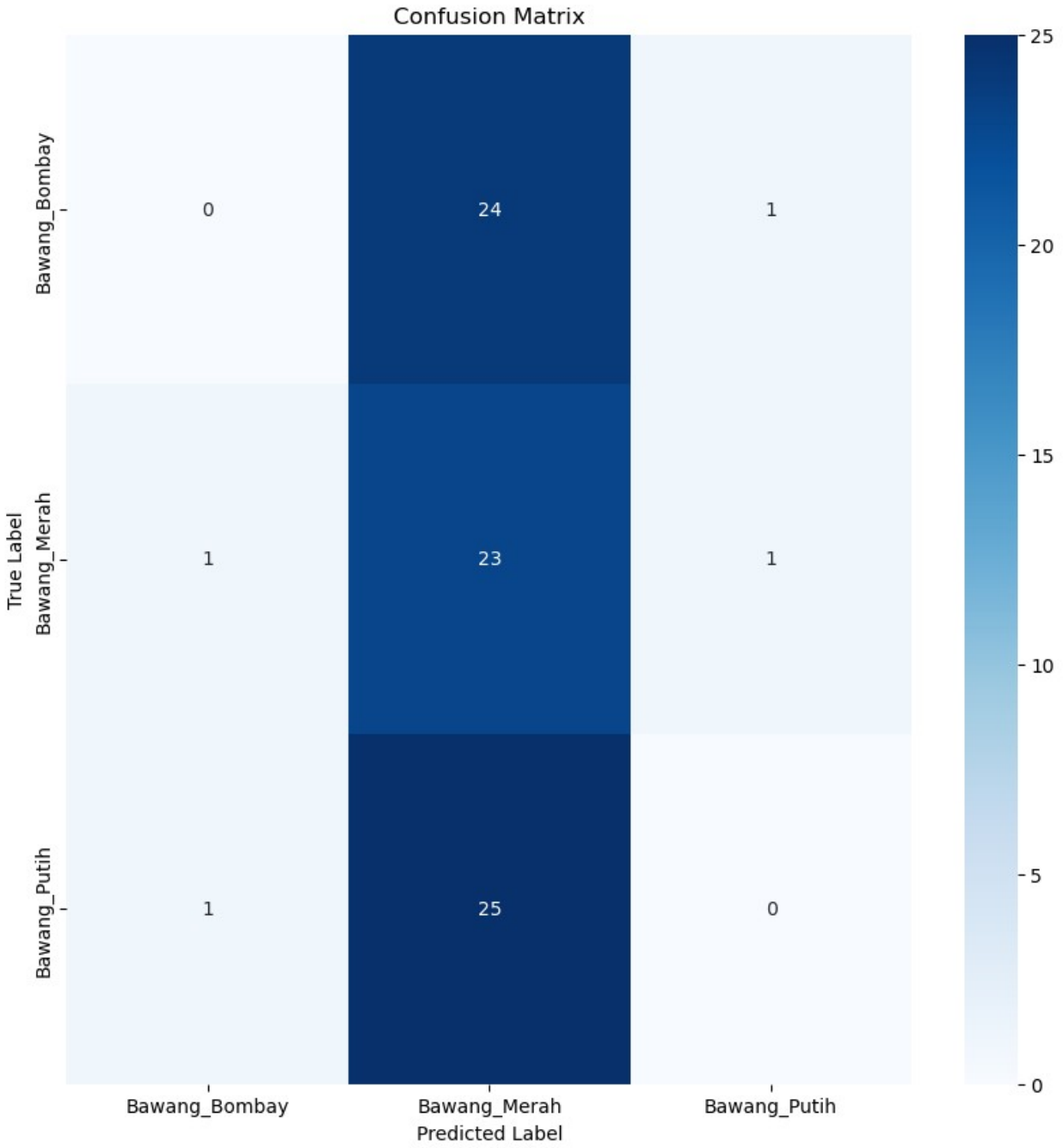
```
# prediksi untuk set data uji
model = load_model(r'D:\Projek UAS PMDPM SHOGUN\
BestModel_GoogleNet_Shogun.h5')
class_names = ['Bawang_Bombay', 'Bawang_Merah', 'Bawang_Putih']

# klasifikasi dataset
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        input_image = tf.keras.utils.load_img(image_path,
target_size=(224, 224))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array,
axis=0)

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

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

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

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

result = classify_images(r'D:\Projek UAS PMDPM SHOGUN\Dataset\test\
Garlic\Garlic_249.jpg', save_path='Garlic_GoogleNet.jpg')
print(result)
```

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

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step
Prediction: Bawang_Merah
Confidence: 33.33%
```

```
Prediksi: Bawang_Merah dengan confidence : 33.33%. Gambar asli
disimpan di Garlic_GoogleNet.jpg.

# memuat dataset uji
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'D:\Projek UAS PMDPM SHOGUN\dataset\test',
    labels='inferred',
    label_mode='categorical',
    image_size=(224, 224),
    batch_size=32
)

# prediksi dataset uji
y_pred = model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)

# mengambil label sebenarnya
true_labels = []
for images, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())
true_labels = tf.convert_to_tensor(true_labels)

# menghitung confusion matrix
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

# menghitung matrix evaluasi
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)
precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=1)

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

# menampilkan confusion matrix
plt.figure(figsize=(10, 10))
sns.heatmap(
    conf_mat,
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=class_names,
    yticklabels=class_names
)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```python
# menampilkan hasil evaluasi
print('Confusion Matrix:\n', conf_mat.numpy())
print('Accuracy:', accuracy.numpy())
print('Precision:', precision.numpy())
print('Recall:', recall.numpy())
print('F1 Score:', f1_score.numpy())
# 0.421
```

```
Found 76 files belonging to 3 classes.
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 150ms/step
```




```
Confusion Matrix:
 [[ 0 24  1]
 [ 1 23  1]
 [ 1 25  0]]
Accuracy: 0.3026315789473684
Precision: [0.         0.31944444 0.        ]
Recall: [0.   0.92 0.  ]
F1 Score: [      nan 0.4742268       nan]
```

MobileNet

Nama Kelompok: Shogun

Anggota Kelompok: (beserta jobdesknya)

1. Yohani Seprini (210711478) mengumpulkan dan menentukan dataset (untuk train validation test split), mengatasi error pada arsitekture model (AlexNet, GoogleNet, MobileNet, Vgg-16), mengerjakan data preparation (mengubah dataset menjadi iterator numpy, mengambil batch dari iterator,normalisasi data dan menampilkan hasil sebelum dan setelah normalisasi, menghitung jumlah batch dalam dataset, menampilkan visualisasi gambar setelah normalisasi), data augmentasi, implementasi data augmentasi menyimpan akurasi dan loss, mengerjakan model deployment, analisis hasil model dan menentukan model terbaik, dan melakukan deployment pada streamlit
2. Marcella Alicia Ndala (220711907) mengerjakan arsitektur model AlexNet, mengerjakan preprocessing data, menampilkan visualisasi data gambar dari dataset, mengerjakan grafik akurasi dan loss AlexNet
3. Mardika Gidion Omega Limbongan (220712025) mengerjakan arsitektur model GoogleNet, menentukan parameter model alexnet, training model dan memantau proses training, impelementasi early stopping dan callbacks, melakukan penyimpanan model setelah training
4. Aprilius Setio Budi Juja (220712045) mengerjakan arsitektur model MobileNet, impelementasi prediksi untuk dataset uji pada semua model (AlexNet, GoogleNet, MobileNet, VGG-16),menghitung dan menampilkan confusion matrix, menghitung evaluasi metrik model (AlexNet, GoogleNet, MobileNet, VGG-16), visualisasi confusion matrix
5. Jawara Theo Christo (220712066) mengerjakan arsitektur model VGG-16, pengujian model dataset uji untuk semua model (AlexNet, GoogleNet, MobileNet, VGG-16), membantu analisis hasil prediksi, analisis kesalahan prediksi, melakukan perbandingan antar model (AlexNet, GoogleNet, MobileNet, VGG-16)

Data Loading

```
# import library
import os
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
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
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from PIL import Image

# direktori dataset
count = 0
dirs = os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\train')
for dir in dirs:
    files = list(os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\
train/' + dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Garlic Folder has 250 Images
Onion Folder has 250 Images
Red_Onion Folder has 250 Images
Images Folder has 750 Images
```

```
# membaca data dari direktori
base_dir = r'D:\Projek UAS PMDPM SHOGUN\Dataset\train'
validation_split = 0.1

# membaca data dari direktori
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(224, 224),
    batch_size=32,
)

# menampilkan class name
class_names = dataset.class_names
print("Class names:", class_names)
```

```
Found 750 files belonging to 3 classes.
Class names: ['Garlic', 'Onion', 'Red_Onion']
```

```
# train validation test split
total_count = len(list(dataset))
val_count = int(total_count * validation_split)
train_count = total_count - val_count
test_count = int(len(dataset) * 0.1)
```

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

train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Data Visualization

```
# menampilkan data gambar dengan paramter jumlah gambar yang
ditampilkan
i = 0
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1) # ukuran gambar
        plt.imshow(images[i].numpy().astype("uint8")) # label gambar
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

Data Preparation

```
# mengubah dataset menjadi iterator numpy
data_iterator = dataset.as_numpy_iterator()
print("data_iterator:", data_iterator)

# mengambil batch berikutnya dari iterator
batch = data_iterator.next()
print("batch:", batch)
```

```
data_iterator:
NumpyIterator(iterator=<tensorflow.python.data.ops.iterator_ops.OwnedI
```

```
terator object at 0x000002197B3CA390>)
batch: (array([[[[ 66.96429  ,  39.964287 ,   9.964286 ],
         [ 65.89286  ,  38.892857 ,   8.892858 ],
         [ 64.82143  ,  37.82143  ,   7.821429 ],
         ...,
         [ 82.82143  ,  52.821426 ,  16.821426 ],
         [ 82.10715  ,  52.107147 ,  16.107147 ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 66.01786  ,  39.017857 ,   9.017858 ],
         [ 64.946434 ,  37.946426 ,   7.9464283],
         [ 63.705994 ,  36.705994 ,   6.7059956],
         ...,
         [ 82.04401  ,  52.044006 ,  16.044004 ],
         [ 82.00574  ,  52.00574  ,  16.00574  ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 68.69643  ,  39.875    ,   8.053572 ],
         [ 67.52742  ,  38.705994 ,   6.8845663],
         [ 65.626915 ,  36.805485 ,   4.984056 ],
         ...,
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ]],

        ...,

        [[128.06625  ,  80.066246 ,  32.06625  ],
         [124.27994  ,  76.38708  ,  28.601364 ],
         [122.04784  ,  75.58355  ,  29.40498  ],
         ...,
         [107.780014 ,  60.780014 ,  14.780015 ],
         [111.85843  ,  65.75128  ,  17.072723 ],
         [115.074615 ,  69.074615 ,  20.074615 ]],

        [[120.210464 ,  72.210464 ,  24.210468 ],
         [120.29848  ,  72.40562  ,  24.6199   ],
         [122.3903   ,  75.92601  ,  29.747442 ],
         ...,
         [112.97258  ,  65.97258  ,  19.97258  ],
         [114.89859  ,  68.79144  ,  20.112888 ],
         [116.93239  ,  70.93239  ,  21.932386 ]],

        [[123.92857  ,  75.92857  ,  27.928572 ],
         [121.78571  ,  73.89286  ,  26.107143 ],
         [119.82143  ,  73.35714  ,  27.178572 ],
         ...,
         [115.64285  ,  68.64285  ,  22.642853 ],
         [116.89285  ,  70.785706 ,  22.107147 ],
         [117.       ,  71.       ,  22.       ]]],
```

```
       [[[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        ...,

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
```

```
         [255.       , 255.       , 255.       ]]],


       [[[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 11.390324 ,  11.390324 ,   1.3903232],
         [ 15.337053 ,  11.337053 ,   0.3370536],
         [ 15.337053 ,  11.337053 ,   0.3370536]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 12.011161 ,  12.011161 ,   2.0111609],
         [ 16.       ,  12.       ,   1.       ],
         [ 16.       ,  12.       ,   1.       ]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 13.630205 ,  13.630205 ,   3.6302047],
         [ 16.685268 ,  12.685268 ,   1.6852679],
         [ 16.685268 ,  12.685268 ,   1.6852679]],

        ...,

        [[100.42539  , 106.42539  , 104.42539  ],
         [112.48364  , 118.48364  , 116.48364  ],
         [110.46183  , 116.7029   , 116.62254  ],
         ...,
         [137.68839  , 138.68839  , 133.68839  ],
         [125.148094 , 127.614136 , 120.88112  ],
         [127.43207  , 130.04425  , 123.23816  ]],

        [[100.64094  , 106.64094  , 104.64094  ],
         [ 93.03762  ,  99.03762  ,  97.03762  ],
         [100.624855 , 106.86593  , 104.96678  ],
         ...,
         [136.88385  , 137.72493  , 132.8044   ],
         [135.82593  , 136.82932  , 130.82762  ],
         [135.20984  , 135.       , 129.60492  ]],

        [[ 97.74758  , 103.74758  , 101.74758  ],
         [102.86773  , 108.86773  , 106.86773  ],
         [114.52398  , 120.76505  , 118.84541  ],
         ...,
         [140.7211   , 140.341    , 136.03105  ],
```

```
         [140.0624   , 137.73647  , 132.73647  ],
         [137.39731  , 135.07138  , 130.07138  ]]],


       ...,


       [[[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        ...,

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [254.48987  , 255.       , 250.48987  ],
         [254.28577  , 255.       , 250.28577  ],
         [253.57153  , 254.28577  , 249.57153  ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
```

```
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        ...,
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ]]],


      [[[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       ...,

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],
```

```
        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]]],


       [[[199.38632  , 206.38632  , 222.38632  ],
         [199.09822  , 206.09822  , 222.09822  ],
         [199.32446  , 206.32446  , 222.32446  ],
         ...,
         [212.31535  , 226.98212  , 232.14873  ],
         [213.00351  , 224.19641  , 230.19641  ],
         [210.86845  , 221.86845  , 227.86845  ]],

        [[199.88217  , 206.88217  , 222.88217  ],
         [197.56616  , 204.56616  , 220.56616  ],
         [196.34805  , 203.34805  , 219.34805  ],
         ...,
         [210.44629  , 225.44629  , 230.44629  ],
         [208.56616  , 222.61974  , 228.09294  ],
         [209.98564  , 223.26788  , 229.12677  ]],

        [[198.12196  , 205.12196  , 221.12196  ],
         [198.24124  , 205.24124  , 221.24124  ],
         [196.18478  , 203.18478  , 219.18478  ],
         ...,
         [209.20535  , 222.20535  , 228.20535  ],
         [209.74106  , 222.74106  , 228.74106  ],
         [207.73212  , 220.73212  , 226.73212  ]],

        ...,

        [[166.30693  , 154.30693  , 166.30693  ],
         [160.22354  , 148.04466  , 160.13411  ],
         [153.68944  , 142.7788   , 155.14476  ],
         ...,
         [240.       , 241.       , 236.       ],
         [239.       , 240.       , 235.       ],
         [239.       , 240.       , 235.       ]],

        [[160.50465  , 156.08504  , 167.55824  ],
         [165.44855  , 158.29518  , 172.87187  ],
         [154.28409  , 144.22205  , 159.9094   ],
         ...,
         [240.6607   , 241.6607   , 236.6607   ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]],
```

```
        [[159.2946   , 155.49103  , 165.2946   ],
         [158.72507  , 150.72507  , 165.52512  ],
         [165.58325  , 156.05634  , 171.5653   ],
         ...,
         [239.       , 240.       , 235.       ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]]], dtype=float32),
array([0, 2, 0, 2, 1, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 0, 0, 2, 2, 2, 2,
0,
       0, 2, 1, 1, 1, 1, 0, 1, 2, 0]))

# normalisasi data dengan membagi nilai piksel dengan 255.0
data = dataset.map(lambda x, y: (x/255.0, y))

# tampil tipe data setelah normalisasi
print("Data type after normalization:
{}".format(dataset.element_spec))
# tampil bentuk data setelah normalisasi
print("Data shape after normalization:
{}".format(dataset.element_spec))
# hitung jumlah batch dalam dataset
print("Jumlah images:", len(dataset))
```

```
Data type after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Data shape after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Jumlah images: 24
```

```
# visualisasi gambar setelah normalisasi
plt.figure(figsize=(10, 10))
for images, labels in data.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy())
        plt.title(class_names[labels[i]])
        plt.axis("off")
plt.show()
```


```
# train validation test split
total_count = len(list(dataset))
val_count = int(total_count * validation_split)
train_count = total_count - val_count
test_count = int(len(dataset) * 0.1)

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

```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Model Architecture

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

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

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

# data augmentation
data_augmentation = Sequential([
    layers.RandomFlip("horizontal_and_vertical", input_shape=(224,
224, 3)),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
    layers.RandomContrast(0.2),
])

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

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


```
# MobileNet
base_model =  tf.keras.applications.MobileNet(
    weights='imagenet',
    include_top=False,
    input_shape=(224, 224, 3)
)

base_model.trainable = True

fine_tune_at = len(base_model.layers) // 3
for layer in base_model.layers[:fine_tune_at]:
```

```
    layer.trainable = False

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

modelMobileNet.compile(
    optimizer=Adam(learning_rate=1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

modelMobileNet.summary()
```

Model: "sequential_13"

Layer (type)	Output Shape	Param #
sequential_12 (Sequential)	(None, 224, 224, 3)	0
rescaling_6 (Rescaling)	(None, 224, 224, 3)	0
mobilenet_1.00_224 (Functional)	(None, 7, 7, 1024)	3,228,864
global_average_pooling2d_6 (GlobalAveragePooling2D)	(None, 1024)	0
dense_12 (Dense)	(None, 128)	131,200

```
| dropout_6 (Dropout)                 | (None, 128)               |
0 |
├─────────────────────────────────────┼───────────────────────────┼──────────
──────┤
| dense_13 (Dense)                    | (None, 3)                 |
387 |
└─────────────────────────────────────┴───────────────────────────┴──────────
──────┘

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

 Trainable params: 3,273,731 (12.49 MB)

 Non-trainable params: 86,720 (338.75 KB)
```

Model Training

```
# training menggunakan iteriasi
early_stopping = EarlyStopping(
    monitor='val_accuracy',
    patience=3,
    mode='max'
)

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

Epoch 1/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 31s 950ms/step - accuracy: 0.3427 - loss:
1.7572 - val_accuracy: 0.2826 - val_loss: 1.9101
Epoch 2/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 20s 906ms/step - accuracy: 0.3671 - loss:
1.3562 - val_accuracy: 0.3043 - val_loss: 1.5002
Epoch 3/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 20s 919ms/step - accuracy: 0.4893 - loss:
1.0673 - val_accuracy: 0.3478 - val_loss: 1.1810
Epoch 4/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 948ms/step - accuracy: 0.5500 - loss:
0.9994 - val_accuracy: 0.4783 - val_loss: 0.9350
Epoch 5/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 20s 925ms/step - accuracy: 0.6308 - loss:
0.8134 - val_accuracy: 0.6957 - val_loss: 0.7355
Epoch 6/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 956ms/step - accuracy: 0.6654 - loss:
0.7227 - val_accuracy: 0.7609 - val_loss: 0.6013
Epoch 7/30
```

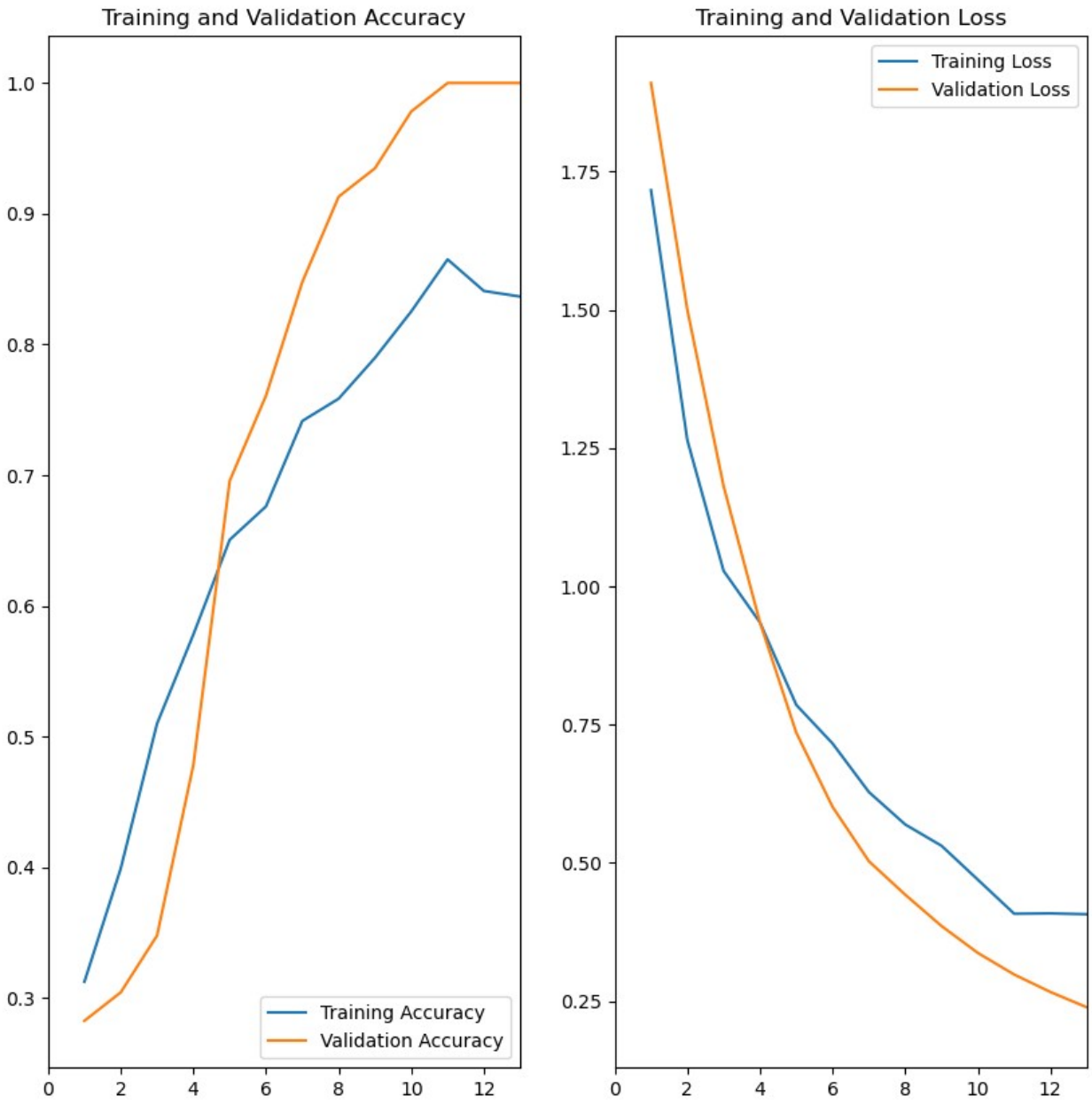
```
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 958ms/step - accuracy: 0.7240 - loss:
0.6482 - val_accuracy: 0.8478 - val_loss: 0.5030
Epoch 8/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 20s 921ms/step - accuracy: 0.7592 - loss:
0.5685 - val_accuracy: 0.9130 - val_loss: 0.4425
Epoch 9/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 961ms/step - accuracy: 0.7802 - loss:
0.5511 - val_accuracy: 0.9348 - val_loss: 0.3859
Epoch 10/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 954ms/step - accuracy: 0.8367 - loss:
0.4538 - val_accuracy: 0.9783 - val_loss: 0.3372
Epoch 11/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 958ms/step - accuracy: 0.8551 - loss:
0.4212 - val_accuracy: 1.0000 - val_loss: 0.2982
Epoch 12/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 972ms/step - accuracy: 0.8580 - loss:
0.3906 - val_accuracy: 1.0000 - val_loss: 0.2665
Epoch 13/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 21s 963ms/step - accuracy: 0.8054 - loss:
0.4601 - val_accuracy: 1.0000 - val_loss: 0.2389
Epoch 14/30
22/22 ━━━━━━━━━━━━━━━━━━━━ 22s 987ms/step - accuracy: 0.8616 - loss:
0.3663 - val_accuracy: 1.0000 - val_loss: 0.2145
```

```
# menyimpan akurasi dan loss
history_df = pd.DataFrame(history.history)
print(history_df)
```

```
    accuracy      loss  val_accuracy  val_loss
0   0.312500  1.716265      0.282609  1.910051
1   0.399148  1.264536      0.304348  1.500232
2   0.509943  1.027880      0.347826  1.181036
3   0.578125  0.934822      0.478261  0.934966
4   0.650568  0.785700      0.695652  0.735509
5   0.676136  0.715880      0.760870  0.601254
6   0.741477  0.628058      0.847826  0.502969
7   0.758523  0.569484      0.913043  0.442533
8   0.789773  0.530914      0.934783  0.385851
9   0.825284  0.469637      0.978261  0.337191
10  0.865057  0.408124      1.000000  0.298159
11  0.840909  0.408773      1.000000  0.266518
12  0.836648  0.407169      1.000000  0.238924
13  0.856534  0.378320      1.000000  0.214519
```

```
# visualisasi akurasi dan loss
ephocs_range = range(1, len(history.history['loss']) + 1)
plt.figure(figsize=(10, 10))

plt.subplot(1, 2, 1)
plt.plot(ephocs_range, history.history['accuracy'], label='Training
```

```
Accuracy')
plt.plot(ephocs_range, history.history['val_accuracy'],
label='Validation Accuracy')
plt.legend(loc='lower right')
plt.xlim(0, 13)
plt.title('Training and Validation Accuracy')

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




```
# menyimpan model
modelMobileNet.save('BestModel_MobileNet_Shogun.h5')
```

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

Model Evaluation

```
# prediksi untuk set data uji
model = load_model(r'D:\Projek UAS PMDPM SHOGUN\
BestModel_MobileNet_Shogun.h5')
class_names = ['Bawang_Bombay', 'Bawang_Merah', 'Bawang_Putih']

# klasifikasi dataset
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        input_image = tf.keras.utils.load_img(image_path,
target_size=(224, 224))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array,
axis=0)

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

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

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

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

result = classify_images(r'D:\Projek UAS PMDPM SHOGUN\Dataset\test\
Onion\Onion_248.jpg', save_path='Onion_MobileNet.jpg')
print(result)
```

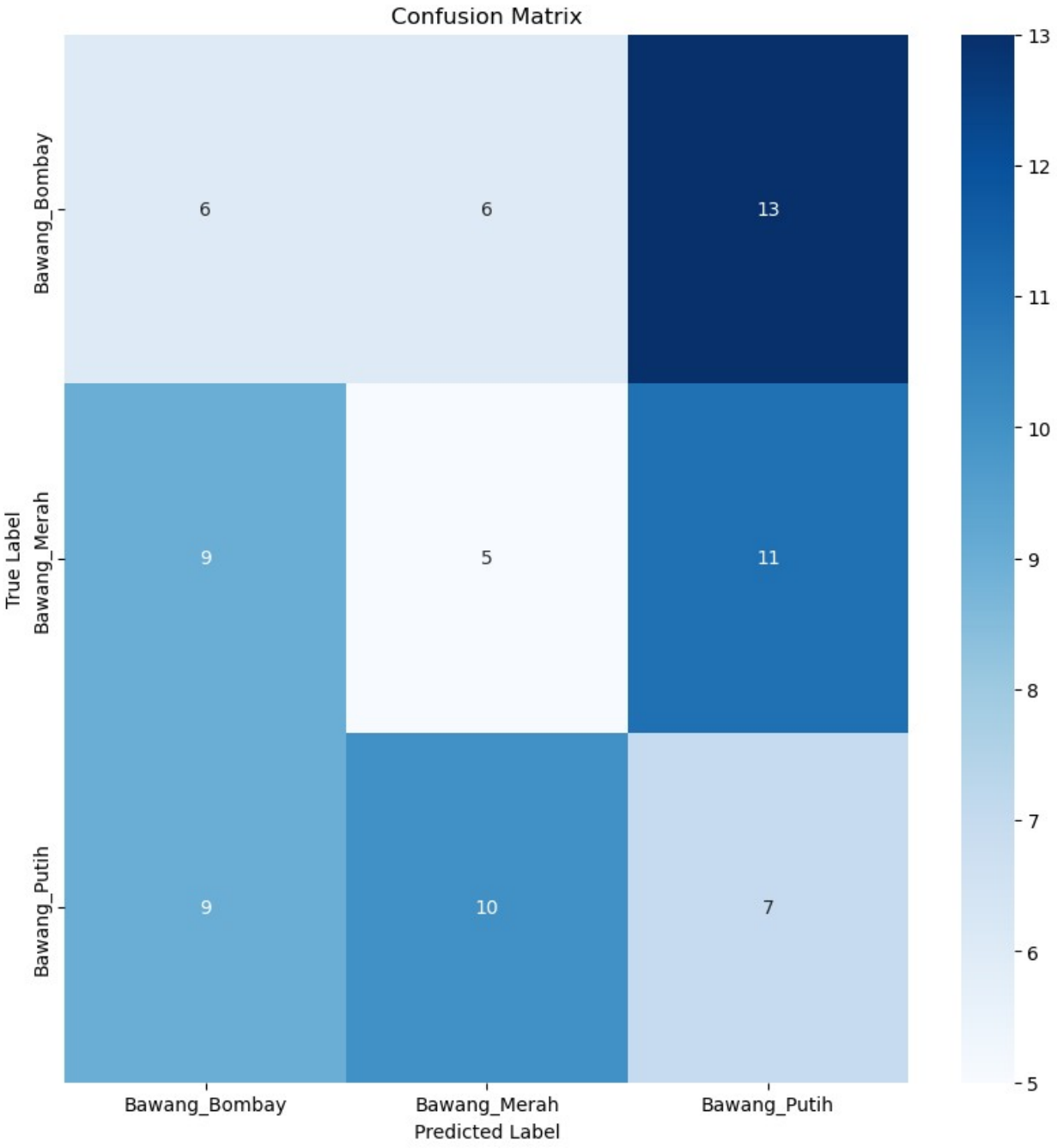
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 598ms/step
Prediction: Bawang_Putih
Confidence: 42.42%
Prediksi: Bawang_Putih dengan confidence : 42.42%. Gambar asli
disimpan di Onion_MobileNet.jpg.
```

```
# memuat dataset uji
mobileNet_model = load_model(r'D:\Projek UAS PMDPM SHOGUN\
BestModel_MobileNet_Shogun.h5')

test_data = tf.keras.preprocessing.image_dataset_from_directory(
```

```
    r'D:\Projek UAS PMDPM SHOGUN\dataset\test',
    labels='inferred',
    label_mode='categorical',
    image_size=(224, 224),
    batch_size=32
)

# prediksi dataset uji
y_pred = mobileNet_model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)

# mengambil label sebenarnya
true_labels = []
for images, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())
true_labels = tf.convert_to_tensor(true_labels)

# menghitung confusion matrix
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

# menghitung matrix evaluasi
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)

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

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

# menampilkan confusion matrix
plt.figure(figsize=(10, 10))
sns.heatmap(conf_mat, annot=True, fmt='d',
cmap='Blues',xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# menampilkan hasil evaluasi
print('Confusion Matrix:\n', conf_mat.numpy())
print('Accuracy:', accuracy.numpy())
print('Precision:', precision.numpy())
print('Recall:', recall.numpy())
print('F1 Score:', f1_score.numpy())
```

```
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 76 files belonging to 3 classes.
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 499ms/step
```




```
Confusion Matrix:
 [[ 6  6 13]
 [ 9  5 11]
 [ 9 10  7]]
Accuracy: 0.23684210526315788
```

```
Precision: [0.25       0.23809524 0.22580645]
Recall: [0.24       0.2        0.26923077]
F1 Score: [0.24489796 0.2173913  0.24561404]
```

VGG-16

Nama Kelompok: Shogun

Anggota Kelompok: (beserta jobdesknya)

1. Yohani Seprini (210711478) mengumpulkan dan menentukan dataset (untuk train validation test split), mengatasi error pada arsitekture model (AlexNet, GoogleNet, MobileNet, Vgg-16), mengerjakan data preparation (mengubah dataset menjadi iterator numpy, mengambil batch dari iterator,normalisasi data dan menampilkan hasil sebelum dan setelah normalisasi, menghitung jumlah batch dalam dataset, menampilkan visualisasi gambar setelah normalisasi), data augmentasi, implementasi data augmentasi menyimpan akurasi dan loss, mengerjakan model deployment, analisis hasil model dan menentukan model terbaik, dan melakukan deployment pada streamlit
2. Marcella Alicia Ndala (220711907) mengerjakan arsitektur model AlexNet, mengerjakan preprocessing data, menampilkan visualisasi data gambar dari dataset, mengerjakan grafik akurasi dan loss AlexNet
3. Mardika Gidion Omega Limbongan (220712025) mengerjakan arsitektur model GoogleNet, menentukan parameter model alexnet, training model dan memantau proses training, impelementasi early stopping dan callbacks, melakukan penyimpanan model setelah training
4. Aprilius Setio Budi Juja (220712045) mengerjakan arsitektur model MobileNet, impelementasi prediksi untuk dataset uji pada semua model (AlexNet, GoogleNet, MobileNet, VGG-16),menghitung dan menampilkan confusion matrix, menghitung evaluasi metrik model (AlexNet, GoogleNet, MobileNet, VGG-16), visualisasi confusion matrix
5. Jawara Theo Christo (220712066) mengerjakan arsitektur model VGG-16, pengujian model dataset uji untuk semua model (AlexNet, GoogleNet, MobileNet, VGG-16), membantu analisis hasil prediksi, analisis kesalahan prediksi, melakukan perbandingan antar model (AlexNet, GoogleNet, MobileNet, VGG-16)

Data Loading

```
# import library
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras._tf_keras.keras.backend as K
import cv2
import os
```

```
import keras
import seaborn as sns

from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model
from keras._tf_keras.keras.models import  Model, load_model
from keras._tf_keras.keras.layers import Input, Dense, Conv2D,
Flatten, MaxPooling2D, AvgPool2D
from keras._tf_keras.keras.layers import Concatenate, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from PIL import Image
from tensorflow.keras.applications import VGG16

# direktori dataset
count = 0
dirs = os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\train')
for dir in dirs:
    files = list(os.listdir(r'D:\Projek UAS PMDPM SHOGUN\Dataset\
train/' + dir))
    print(dir + ' Folder has ' + str(len(files)) + ' Images')
    count = count + len(files)
print('Images Folder has ' + str(count) + ' Images')
```

```
Garlic Folder has 250 Images
Onion Folder has 250 Images
Red_Onion Folder has 250 Images
Images Folder has 750 Images
```

```
# membaca data dari direktori
base_dir = r'D:\Projek UAS PMDPM SHOGUN\Dataset\train'
validation_split = 0.1

# membuat dataset berupa parameter fungsi
dataset = tf.keras.utils.image_dataset_from_directory(
    base_dir,
    seed=123,
    image_size=(224, 224),
    batch_size=32,
    shuffle=True,
)

# menampilkan class name
class_names = dataset.class_names
print("Class names:", class_names)
```

```
Found 750 files belonging to 3 classes.
Class names: ['Garlic', 'Onion', 'Red_Onion']
```

```
# train validation test split
total_count = len(list(dataset))
```

```
val_count = int(total_count * validation_split)
train_count = total_count - val_count
test_count = int(len(dataset) * 0.1)

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

train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Data Visualization

```
# menampilkan data gambar dengan paramter jumlah gambar yang
ditampilkan
i = 0
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1) # ukuran gambar
        plt.imshow(images[i].numpy().astype("uint8")) # label gambar
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

Data Preparation

```
# mengubah dataset menjadi iterator numpy
data_iterator = dataset.as_numpy_iterator()
print("data_iterator:", data_iterator)

# mengambil batch berikutnya dari iterator
batch = data_iterator.next()
print("batch:", batch)
```

```
data_iterator:
NumpyIterator(iterator=<tensorflow.python.data.ops.iterator_ops.OwnedI
```

```
terator object at 0x00000247A3A255D0>)
batch: (array([[[[ 66.96429  ,  39.964287 ,   9.964286 ],
         [ 65.89286  ,  38.892857 ,   8.892858 ],
         [ 64.82143  ,  37.82143  ,   7.821429 ],
         ...,
         [ 82.82143  ,  52.821426 ,  16.821426 ],
         [ 82.10715  ,  52.107147 ,  16.107147 ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 66.01786  ,  39.017857 ,   9.017858 ],
         [ 64.946434 ,  37.946426 ,   7.9464283],
         [ 63.705994 ,  36.705994 ,   6.7059956],
         ...,
         [ 82.04401  ,  52.044006 ,  16.044004 ],
         [ 82.00574  ,  52.00574  ,  16.00574  ],
         [ 82.       ,  52.       ,  16.       ]],

        [[ 68.69643  ,  39.875    ,   8.053572 ],
         [ 67.52742  ,  38.705994 ,   6.8845663],
         [ 65.626915 ,  36.805485 ,   4.984056 ],
         ...,
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ],
         [ 80.17857  ,  50.17857  ,  16.       ]],

        ...,

        [[128.06625  ,  80.066246 ,  32.06625  ],
         [124.27994  ,  76.38708  ,  28.601364 ],
         [122.04784  ,  75.58355  ,  29.40498  ],
         ...,
         [107.780014 ,  60.780014 ,  14.780015 ],
         [111.85843  ,  65.75128  ,  17.072723 ],
         [115.074615 ,  69.074615 ,  20.074615 ]],

        [[120.210464 ,  72.210464 ,  24.210468 ],
         [120.29848  ,  72.40562  ,  24.6199   ],
         [122.3903   ,  75.92601  ,  29.747442 ],
         ...,
         [112.97258  ,  65.97258  ,  19.97258  ],
         [114.89859  ,  68.79144  ,  20.112888 ],
         [116.93239  ,  70.93239  ,  21.932386 ]],

        [[123.92857  ,  75.92857  ,  27.928572 ],
         [121.78571  ,  73.89286  ,  26.107143 ],
         [119.82143  ,  73.35714  ,  27.178572 ],
         ...,
         [115.64285  ,  68.64285  ,  22.642853 ],
         [116.89285  ,  70.785706 ,  22.107147 ],
         [117.       ,  71.       ,  22.       ]]],
```

```
       [[[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        ...,

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]],

        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
```

```
         [255.       , 255.       , 255.       ]]],


       [[[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 11.390324 ,  11.390324 ,   1.3903232],
         [ 15.337053 ,  11.337053 ,   0.3370536],
         [ 15.337053 ,  11.337053 ,   0.3370536]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 12.011161 ,  12.011161 ,   2.0111609],
         [ 16.       ,  12.       ,   1.       ],
         [ 16.       ,  12.       ,   1.       ]],

        [[  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         [  0.       ,   0.       ,   0.       ],
         ...,
         [ 13.630205 ,  13.630205 ,   3.6302047],
         [ 16.685268 ,  12.685268 ,   1.6852679],
         [ 16.685268 ,  12.685268 ,   1.6852679]],

        ...,

        [[100.42539  , 106.42539  , 104.42539  ],
         [112.48364  , 118.48364  , 116.48364  ],
         [110.46183  , 116.7029   , 116.62254  ],
         ...,
         [137.68839  , 138.68839  , 133.68839  ],
         [125.148094 , 127.614136 , 120.88112  ],
         [127.43207  , 130.04425  , 123.23816  ]],

        [[100.64094  , 106.64094  , 104.64094  ],
         [ 93.03762  ,  99.03762  ,  97.03762  ],
         [100.624855 , 106.86593  , 104.96678  ],
         ...,
         [136.88385  , 137.72493  , 132.8044   ],
         [135.82593  , 136.82932  , 130.82762  ],
         [135.20984  , 135.       , 129.60492  ]],

        [[ 97.74758  , 103.74758  , 101.74758  ],
         [102.86773  , 108.86773  , 106.86773  ],
         [114.52398  , 120.76505  , 118.84541  ],
         ...,
         [140.7211   , 140.341    , 136.03105  ],
```

```
         [140.0624   , 137.73647  , 132.73647  ],
         [137.39731  , 135.07138  , 130.07138  ]]],


       ...,


       [[[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        ...,

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [254.48987  , 255.       , 250.48987  ],
         [254.28577  , 255.       , 250.28577  ],
         [253.57153  , 254.28577  , 249.57153  ]],

        [[255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         ...,
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ],
         [255.       , 255.       , 251.       ]],

        [[255.       , 255.       , 251.       ],
```

```
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        ...,
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ],
        [255.       , 255.       , 251.       ]]],


      [[[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       ...,

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],

       [[255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        ...,
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ],
        [255.       , 255.       , 255.       ]],
```

```
        [[255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         ...,
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ],
         [255.       , 255.       , 255.       ]]],


       [[[199.38632  , 206.38632  , 222.38632  ],
         [199.09822  , 206.09822  , 222.09822  ],
         [199.32446  , 206.32446  , 222.32446  ],
         ...,
         [212.31535  , 226.98212  , 232.14873  ],
         [213.00351  , 224.19641  , 230.19641  ],
         [210.86845  , 221.86845  , 227.86845  ]],

        [[199.88217  , 206.88217  , 222.88217  ],
         [197.56616  , 204.56616  , 220.56616  ],
         [196.34805  , 203.34805  , 219.34805  ],
         ...,
         [210.44629  , 225.44629  , 230.44629  ],
         [208.56616  , 222.61974  , 228.09294  ],
         [209.98564  , 223.26788  , 229.12677  ]],

        [[198.12196  , 205.12196  , 221.12196  ],
         [198.24124  , 205.24124  , 221.24124  ],
         [196.18478  , 203.18478  , 219.18478  ],
         ...,
         [209.20535  , 222.20535  , 228.20535  ],
         [209.74106  , 222.74106  , 228.74106  ],
         [207.73212  , 220.73212  , 226.73212  ]],

        ...,

        [[166.30693  , 154.30693  , 166.30693  ],
         [160.22354  , 148.04466  , 160.13411  ],
         [153.68944  , 142.7788   , 155.14476  ],
         ...,
         [240.       , 241.       , 236.       ],
         [239.       , 240.       , 235.       ],
         [239.       , 240.       , 235.       ]],

        [[160.50465  , 156.08504  , 167.55824  ],
         [165.44855  , 158.29518  , 172.87187  ],
         [154.28409  , 144.22205  , 159.9094   ],
         ...,
         [240.6607   , 241.6607   , 236.6607   ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]],
```

```
        [[159.2946   , 155.49103  , 165.2946   ],
         [158.72507  , 150.72507  , 165.52512  ],
         [165.58325  , 156.05634  , 171.5653   ],
         ...,
         [239.       , 240.       , 235.       ],
         [240.       , 241.       , 236.       ],
         [240.       , 241.       , 236.       ]]], dtype=float32),
array([0, 2, 0, 2, 1, 2, 1, 2, 1, 0, 1, 1, 2, 1, 1, 0, 0, 2, 2, 2, 2,
0,
       0, 2, 1, 1, 1, 1, 0, 1, 2, 0]))
```

```
# normalisasi data dengan membagi nilai piksel dengan 255.0
data = dataset.map(lambda x, y: (x/255.0, y))

# tampil tipe data setelah normalisasi
print("Data type after normalization:
{}".format(dataset.element_spec))
# tampil bentuk data setelah normalisasi
print("Data shape after normalization:
{}".format(dataset.element_spec))
# hitung jumlah batch dalam dataset
print("Jumlah images:", len(dataset))
```

```
Data type after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Data shape after normalization: (TensorSpec(shape=(None, 224, 224, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))
Jumlah images: 24
```

```
# visualisasi gambar setelah normalisasi
plt.figure(figsize=(10, 10))
for images, labels in data.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1) # ukuran gambar
        plt.imshow(images[i].numpy()) # label gambar
        plt.title(class_names[labels[i]])
        plt.axis("off")
plt.show()
```


```
# train validation test split
total_count = len(list(dataset))
val_count = int(total_count * validation_split)
train_count = total_count - val_count
test_count = int(len(dataset) * 0.1)

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

```
train_ds = dataset.take(train_count)
val_ds = dataset.skip(train_count).take(val_count)
test_ds = dataset.skip(train_count + val_count).take(test_count)
```

```
Total images: 24
Train images: 22
Validation images: 2
Test images: 2
```

Model Architecture

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

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

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

# data augmentation
data_augmentation = Sequential([
    layers.RandomFlip("horizontal_and_vertical", input_shape=(224,
224, 3)),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
    layers.RandomContrast(0.2),
    layers.RandomTranslation(0.1, 0.1),
])

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

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


```
# VGG16
base_model =  tf.keras.applications.VGG16(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet'
)

base_model.trainable = False

modelVGG16 = Sequential([
    data_augmentation,
```

```
    base_model,
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(class_names), activation='softmax')
])

# compile model
modelVGG16.compile(
    optimizer=Adam(learning_rate=1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

modelVGG16.summary()
```

```
Model: "sequential_5"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ sequential_4 (Sequential)            │ (None, 224, 224, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ vgg16 (Functional)                   │ (None, 7, 7, 512)           │
14,714,688 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten_2 (Flatten)                  │ (None, 25088)               │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_4 (Dense)                      │ (None, 128)                 │
3,211,392 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_2 (Dropout)                  │ (None, 128)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_5 (Dense)                      │ (None, 3)                   │
387 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
```

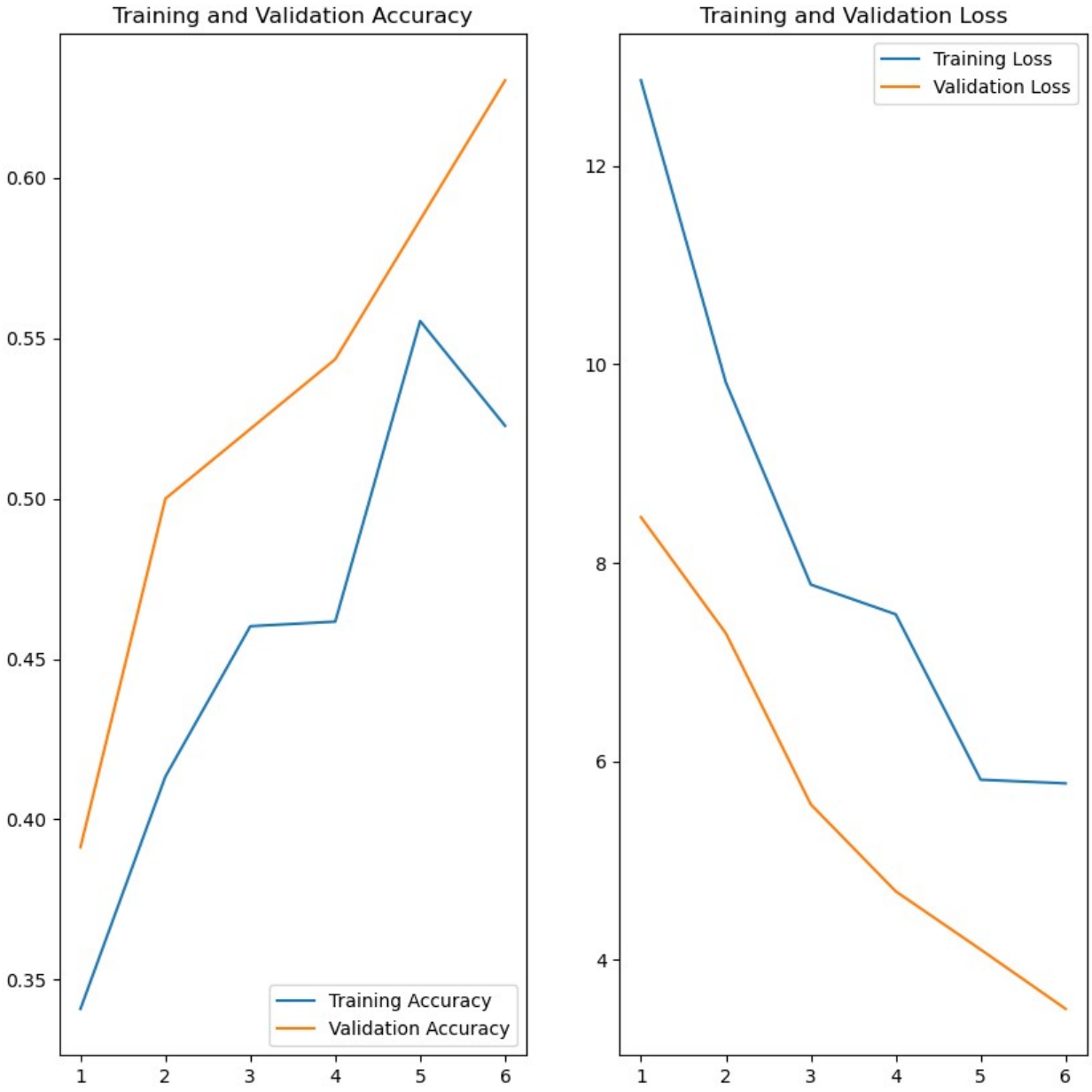
```
 Total params: 17,926,467 (68.38 MB)
```

```
 Trainable params: 3,211,779 (12.25 MB)

 Non-trainable params: 14,714,688 (56.13 MB)
```

Model Training

```
# training menggunakan iteriasi
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    mode='max',
)

history = modelVGG16.fit(
    train_ds,
    validation_data=val_ds,
    epochs=100,
    callbacks=[early_stopping]
)

Epoch 1/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 84s 4s/step - accuracy: 0.3239 - loss:
13.9696 - val_accuracy: 0.3913 - val_loss: 8.4598
Epoch 2/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 92s 4s/step - accuracy: 0.3770 - loss:
10.9986 - val_accuracy: 0.5000 - val_loss: 7.2891
Epoch 3/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 101s 5s/step - accuracy: 0.4461 - loss:
8.4694 - val_accuracy: 0.5217 - val_loss: 5.5634
Epoch 4/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 94s 4s/step - accuracy: 0.4655 - loss:
7.1674 - val_accuracy: 0.5435 - val_loss: 4.6869
Epoch 5/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 88s 4s/step - accuracy: 0.5451 - loss:
5.5841 - val_accuracy: 0.5870 - val_loss: 4.1001
Epoch 6/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 168s 8s/step - accuracy: 0.5465 - loss:
5.3760 - val_accuracy: 0.6304 - val_loss: 3.5018

# menyimpan akurasi dan loss
history_df = pd.DataFrame(history.history)
print(history_df)

   accuracy       loss  val_accuracy  val_loss
0  0.340909  12.864499      0.391304  8.459750
1  0.413352   9.817343      0.500000  7.289078
2  0.460227   7.780037      0.521739  5.563440
3  0.461648   7.480195      0.543478  4.686926
4  0.555398   5.812238      0.586957  4.100128
5  0.522727   5.775218      0.630435  3.501842
```

```
# visualisasi akurasi dan loss
ephocs_range = range(1, len(history.history['accuracy']) + 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()
```




```
# menyimpan model
modelVGG16.save('BestModel_VGG16_Shogun.h5')
```

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

Model Evaluation

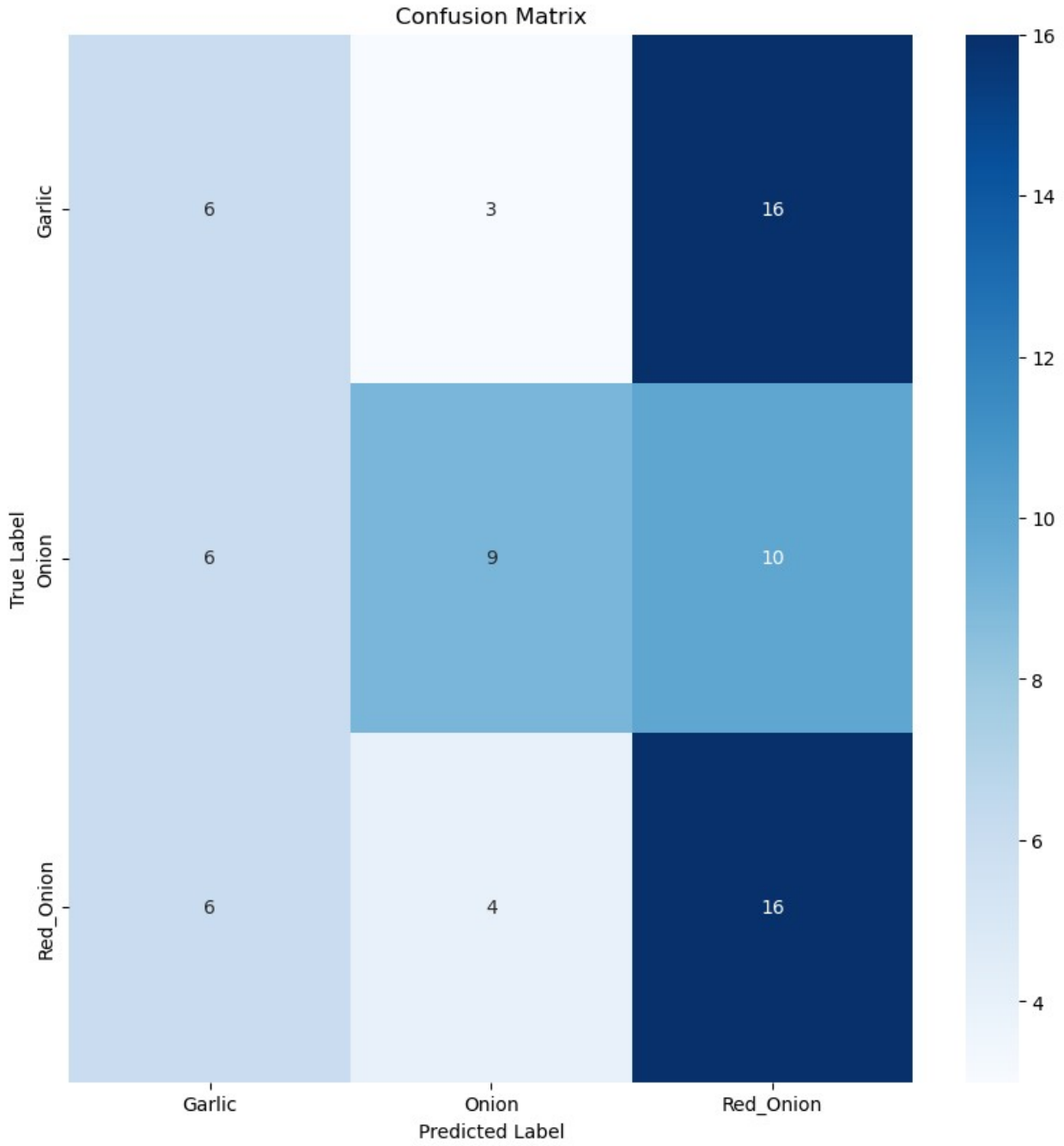
```
# prediksi untuk set data uji
model = load_model('D:\Projek UAS PMDPM SHOGUN\
BestModel_VGG16_Shogun.h5')

# klasifikasi dataset
def classify_images(image_path, save_path='predicted_image.jpg'):
    try:
        input_image = tf.keras.utils.load_img(image_path,
target_size=(224, 224))
        input_image_array = tf.keras.utils.img_to_array(input_image)
        input_image_exp_dim = tf.expand_dims(input_image_array,
axis=0)

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

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

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

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

result = classify_images(r'D:\Projek UAS PMDPM SHOGUN\Dataset\test\
Red_Onion\Red_Onion_250.jpg', save_path='Red_onion.jpg')
print(result)
```

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

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 422ms/step
Prediction: Red_Onion
Confidence: 57.61%
Prediksi: Red_Onion dengan confidence : 57.61%. Gambar asli disimpan
di Red_onion.jpg.
```

```
# memuat dataset uji
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    r'D:\Projek UAS PMDPM SHOGUN\dataset\test',
    labels='inferred',
    label_mode='categorical',
    image_size=(224, 224),
```

```
    batch_size=32
)

# prediksi dataset uji
y_pred = model.predict(test_data)
y_pred_class = tf.argmax(y_pred, axis=1)

# mengambil label sebenarnya
true_labels = []
for images, labels in test_data:
    true_labels.extend(tf.argmax(labels, axis=1).numpy())
true_labels = tf.convert_to_tensor(true_labels)

# menghitung confusion matrix
conf_mat = tf.math.confusion_matrix(true_labels, y_pred_class)

# menghitung matrix evaluasi
accuracy = tf.reduce_sum(tf.linalg.diag_part(conf_mat)) /
tf.reduce_sum(conf_mat)
precision = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=0)
recall = tf.linalg.diag_part(conf_mat) / tf.reduce_sum(conf_mat,
axis=1)

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

# menampilkan confusion matrix
plt.figure(figsize=(10, 10))
sns.heatmap(
    conf_mat,
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=class_names,
    yticklabels=class_names
)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# menampilkan hasil evaluasi
print('Confusion Matrix:\n', conf_mat.numpy())
print('Accuracy:', accuracy.numpy())
print('Precision:', precision.numpy())
print('Recall:', recall.numpy())
print('F1 Score:', f1_score.numpy())
```

```
Found 76 files belonging to 3 classes.
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 3s/step
```




```
Confusion Matrix:
 [[ 6  3 16]
 [ 6  9 10]
 [ 6  4 16]]
Accuracy: 0.40789473684210525
Precision: [0.33333333 0.5625     0.38095238]
Recall: [0.24       0.36       0.61538462]
F1 Score: [0.27906977 0.43902439 0.47058824]
```

MainStreamlit_A_Shogun.py

```python
# dikerjakan oleh: Yohani Seprini (210711478)

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

model = load_model(r"D:\Projek UAS PMDPM SHOGUN\BestModel_VGG16_Shogun.h5")
class_names = ['Onion', 'Red_Onion', 'Garlic']

st.markdown("""
    <style>
        body {
            background-color: #F5F5F5; /* Light grey for modern theme */
            color: #333333; /* Dark grey text */
            font-family: "Arial", sans-serif;
        }
        .title {
            color: #0C0C0C;
            text-align: center;
            margin-bottom: 30px;
        }
        .prediction-box {
            border: 2px solid #9E9E9E;
            border-radius: 10px;
            padding: 20px;
            margin-top: 20px;
            background-color: #FFFFFF;
        }
        .upload-container {
            border: 1px dashed #0C0C0C;
            padding: 15px;
            border-radius: 10px;
            text-align: center;
        }
        .file-name {
            font-weight: bold;
            color: #333333;
        }
    </style>
""", unsafe_allow_html=True)

def classify_image(image):
    try:
        input_image = image.resize((224, 224))
        input_image_array = np.array(input_image) / 255.0
        input_image_array_exp_dim = np.expand_dims(input_image_array, axis=0)
```

```
49
50          predictions = model.predict(input_image_array_exp_dim)
51          result = tf.nn.softmax(predictions[0])
52
53          class_idx = np.argmax(result)
54          confidence_scores = result.numpy()
55          return class_names[class_idx], confidence_scores
56      except Exception as e:
57          return "Error", str(e)
58
59  def custom_progress_bar(confidence, colors):
60      progress_html = f"""
61      <div style="border: 1px solid #ddd; border-radius: 5px; overflow: hidden; width: 100%;
    font-size: 14px; display: flex;">
62          <div style="width: {confidence[0] * 100:.2f}%; background: {colors[0]}; color: white;
    text-align: center; height: 24px;">
63              {confidence[0] * 100:.2f}%
64          </div>
65          <div style="width: {confidence[1] * 100:.2f}%; background: {colors[1]}; color: white;
    text-align: center; height: 24px;">
66              {confidence[1] * 100:.2f}%
67          </div>
68          <div style="width: {confidence[2] * 100:.2f}%; background: {colors[2]}; color: white;
    text-align: center; height: 24px;">
69              {confidence[2] * 100:.2f}%
70          </div>
71      </div>
72      """
73      st.markdown(progress_html, unsafe_allow_html=True)
74
75  st.markdown("<h1 class='title'>🧅 Prediksi Jenis Bawang - Kelompok Shogun</h1>",
    unsafe_allow_html=True)
76  st.markdown("""
77  <div class="upload-container">
78      <p>📤 <strong>Unggah Gambar</strong> (format: jpg, jpeg, png)</p>
79  </div>
80  """, unsafe_allow_html=True)
81
82  uploaded_files = st.file_uploader("", type=["jpg", "jpeg", "png"], accept_multiple_files=True)
83
84  if st.button("🔍 **Prediksi**"):
85      if uploaded_files:
86          st.write("### 📝 Hasil Prediksi:")
87          for uploaded_file in uploaded_files:
88              try:
89                  image = Image.open(uploaded_file)
90                  st.image(image, caption=uploaded_file.name, use_column_width=True)
91                  label, confidence = classify_image(image)
92                  if label != "Error":
93                      colors = ["#FFCA28", "#E53935", "#007BFF"]
94                      st.markdown(f"""
```

```
95                      <div class='prediction-box'>
96                          <p class='file-name'>🖼️ <strong>File Name:</strong>
{uploaded_file.name}</p>
97                          <h4 style='color: {colors[class_names.index(label)]};'>🔮
<strong>Prediction:</strong> {label}</h4>
98                          <p><strong>Confidence Scores:</strong></p>
99                          <ul>
100                             <li>Onion: {confidence[0] * 100:.2f}%</li>
101                             <li>Red Onion: {confidence[1] * 100:.2f}%</li>
102                             <li>Garlic: {confidence[2] * 100:.2f}%</li>
103                         </ul>
104                     </div>
105                     """, unsafe_allow_html=True)
106                     custom_progress_bar(confidence, colors)
107                 else:
108                     st.write(f"❌ Kesalahan saat memproses gambar: {uploaded_file.name}:
{confidence}")
109             except Exception as e:
110                 st.write(f"❌ Error: Tidak dapat memproses file {uploaded_file.name}
({str(e)})")
111     else:
112         st.write("⚠️ Silakan unggah setidaknya satu gambar untuk diprediksi.")
113
```