ANGGOTA KELOMPOK

- Marvel Ravindra Dioputra 2200481
- Ravindra Maulana Sahman 2108724
- Rifanny Lysara Annastasya 2200163
- Revana Faliha Salma 2202869
- Tia Ifania Nugrahaningtyas 2202339

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

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from google.colab import drive

! chmod 600 /content/kaggle.json
#Import dataset dari kaggle
! KAGGLE_CONFIG_DIR=/content/ kaggle datasets download -d tiaifania/kelompok4-datasetimagere

# jika ingin di run harus memasukkan kaggle.json ke folder content
```

```
    kelompok4-datasetimagerecognition.zip: Skipping, found more recently modified local copy
```

```
! unzip kelompok4-datasetimagerecognition
```

```
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (13).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (14).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (15).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (16).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (17).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (18).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (19).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (2).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (20).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (21).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (22).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (23).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (24).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (25).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (26).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (27).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (28).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (29).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (3).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (30).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (31).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (32).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (33).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (34).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (35).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (36).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (37).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (38).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (39).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (4).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (40).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (41).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (5).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (6).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (7).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (8).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images (9).jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/images.jpeg
  inflating: Kelompok4_DatasetImageRecognition/Kendaraan_Mobil/th.jpeg
```

```
#dataset_url = "/content/Kelompok4_Dataset/Kelompok4_Dataset/Kelompok4_Dataset/Kendaraan_Mob
import pathlib
base_dir = "/content/Kelompok4_DatasetImageRecognition"
print(base_dir)
```

```
/content/Kelompok4_DatasetImageRecognition
```

```
print(type(base_dir))
```

```
<class 'str'>
```

```
batch_size = 32
img_height = 180
img_width = 180


# train
# mengambil data secara random



train_ds = tf.keras.utils.image_dataset_from_directory(
  base_dir,
  validation_split=0.3,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
    Found 1356 files belonging to 2 classes.
    Using 950 files for training.
```

```
train_ds
```

```
    <_PrefetchDataset element_spec=(TensorSpec(shape=(None, 180, 180, 3), dtype=tf.float32,
    name=None), TensorSpec(shape=(None,), dtype=tf.int32, name=None))>
```

```
# validasi
#mengambil gambar dari database secara random untuk digunakan dalam val_ds

val_ds = tf.keras.utils.image_dataset_from_directory(
  base_dir,
  validation_split=0.3,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
    Found 1356 files belonging to 2 classes.
    Using 406 files for validation.
```

```
class_names = train_ds.class_names
print(class_names)
```

```
    ['Kendaraan_Bukan_Mobil', 'Kendaraan_Mobil']
```

```
# lihat dataset training
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(2):
    ax = plt.subplot(2, 1, i + 1) # 3 baris, 3 kolom
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

```
# 32 per batch, 180x180 pixel, warna 3 (RGB)
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

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

```
AUTOTUNE = tf.data.AUTOTUNE

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

```
# normalisasi
normalization_layer = layers.Rescaling(1./255)
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# nilai dari [0 sd 255] menjadi [0 sd 1]
print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```

```
num_classes = len(train_ds)

model = tf.keras.models.Sequential([
 layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
 layers.Conv2D(16, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Conv2D(32, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Conv2D(64, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Flatten(),
 layers.Dense(128, activation='relu'),
 layers.Dense(1, activation='sigmoid')
])
```

```
# menggunakan binary karena dataset hanya terdiri dari 2 kelas (kalau lebih dari 2 kelas mem
model.compile(optimizer='adam',
            loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
            metrics=['accuracy'])
```

```
model.summary()
```

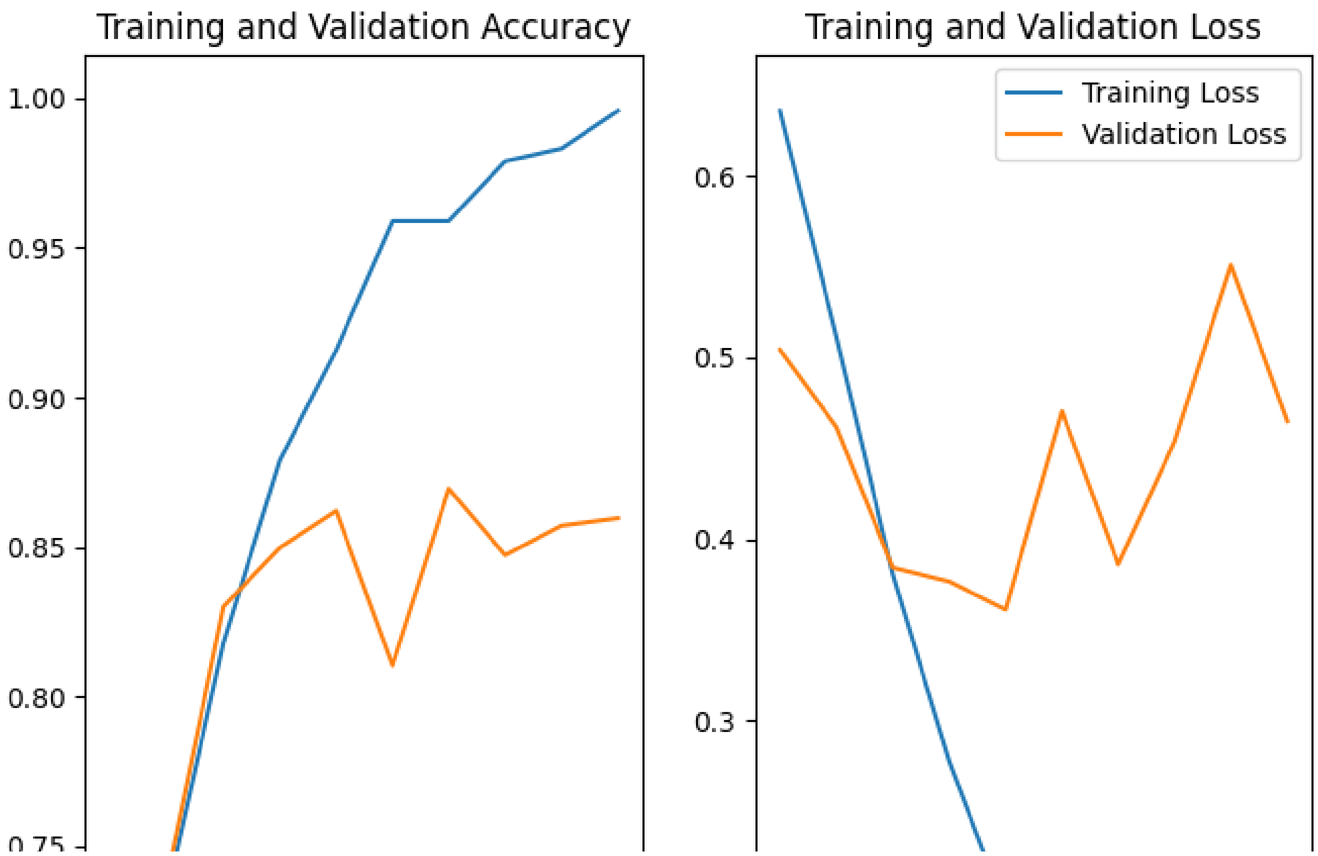
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_3 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d_3 (Conv2D)           (None, 180, 180, 16)      448       
                                                                 
 max_pooling2d_3 (MaxPoolin  (None, 90, 90, 16)        0         
 g2D)                                                            
                                                                 
 conv2d_4 (Conv2D)           (None, 90, 90, 32)        4640      
                                                                 
 max_pooling2d_4 (MaxPoolin  (None, 45, 45, 32)        0         
 g2D)                                                            
                                                                 
 conv2d_5 (Conv2D)           (None, 45, 45, 64)        18496     
                                                                 
 max_pooling2d_5 (MaxPoolin  (None, 22, 22, 64)        0         
 g2D)                                                            
                                                                 
 flatten_1 (Flatten)         (None, 30976)             0         
                                                                 
 dense_2 (Dense)             (None, 128)               3965056   
                                                                 
 dense_3 (Dense)             (None, 1)                 129       
                                                                 
=================================================================
Total params: 3988769 (15.22 MB)
Trainable params: 3988769 (15.22 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
#training data
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/10
30/30 [==============================] - 4s 97ms/step - loss: 0.6359 - accuracy: 0.6253
Epoch 2/10
30/30 [==============================] - 1s 24ms/step - loss: 0.5113 - accuracy: 0.7337
```

```
Epoch 3/10
30/30 [==============================] - 1s 22ms/step - loss: 0.3805 - accuracy: 0.8179
Epoch 4/10
30/30 [==============================] - 1s 23ms/step - loss: 0.2776 - accuracy: 0.8789
Epoch 5/10
30/30 [==============================] - 1s 24ms/step - loss: 0.1955 - accuracy: 0.9158
Epoch 6/10
30/30 [==============================] - 1s 25ms/step - loss: 0.1169 - accuracy: 0.9589
Epoch 7/10
30/30 [==============================] - 1s 26ms/step - loss: 0.1241 - accuracy: 0.9589
Epoch 8/10
30/30 [==============================] - 1s 41ms/step - loss: 0.0822 - accuracy: 0.9789
Epoch 9/10
30/30 [==============================] - 1s 27ms/step - loss: 0.0445 - accuracy: 0.9832
Epoch 10/10
30/30 [==============================] - 1s 27ms/step - loss: 0.0253 - accuracy: 0.9958
```

```
#menampilkan plot
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(epochs)
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



```
# data augmentasi agar mengurangi overfit
data_augmentation = keras.Sequential(
 [
   layers.RandomFlip("horizontal",input_shape=(img_height,img_width,3)),
   layers.RandomRotation(0.1),
   layers.RandomZoom(0.1),
 ]
)
```

```
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
 for i in range(2):
   augmented_images = data_augmentation(images)
   ax = plt.subplot(1, 2, i + 1)
   plt.imshow(augmented_images[0].numpy().astype("uint8"))
   plt.axis("off")
```

```
model = Sequential([
 data_augmentation,
 layers.Rescaling(1./255),
 layers.Conv2D(64, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Conv2D(128, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Conv2D(256, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Conv2D(512, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Dropout(0.2),
 layers.Flatten(),
 #layers.Dense(128, activation='relu'),
 layers.Dense(1, activation='sigmoid')
])
```

```
model.compile(optimizer='adam',
            loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
            metrics=['accuracy'])
epochs = 15
history = model.fit(
 train_ds,
 validation_data=val_ds,
 epochs=epochs
)
```

```
    Epoch 1/15
    30/30 [==============================] - 11s 188ms/step - loss: 0.6923 - accuracy: 0.548
    Epoch 2/15
    30/30 [==============================] - 4s 127ms/step - loss: 0.5937 - accuracy: 0.6958
    Epoch 3/15
    30/30 [==============================] - 4s 125ms/step - loss: 0.5146 - accuracy: 0.7484
    Epoch 4/15
    30/30 [==============================] - 4s 128ms/step - loss: 0.4857 - accuracy: 0.7547
    Epoch 5/15
    30/30 [==============================] - 4s 123ms/step - loss: 0.4903 - accuracy: 0.7663
```

```
Epoch 6/15
30/30 [==============================] - 4s 124ms/step - loss: 0.4376 - accuracy: 0.8074
Epoch 7/15
30/30 [==============================] - 4s 126ms/step - loss: 0.4338 - accuracy: 0.7947
Epoch 8/15
30/30 [==============================] - 4s 124ms/step - loss: 0.3525 - accuracy: 0.8432
Epoch 9/15
30/30 [==============================] - 4s 125ms/step - loss: 0.3461 - accuracy: 0.8389
Epoch 10/15
30/30 [==============================] - 4s 126ms/step - loss: 0.3345 - accuracy: 0.8611
Epoch 11/15
30/30 [==============================] - 4s 126ms/step - loss: 0.3319 - accuracy: 0.8579
Epoch 12/15
30/30 [==============================] - 4s 125ms/step - loss: 0.3182 - accuracy: 0.8674
Epoch 13/15
30/30 [==============================] - 4s 127ms/step - loss: 0.3020 - accuracy: 0.8600
Epoch 14/15
30/30 [==============================] - 4s 126ms/step - loss: 0.3166 - accuracy: 0.8674
Epoch 15/15
30/30 [==============================] - 4s 125ms/step - loss: 0.2746 - accuracy: 0.8726
```

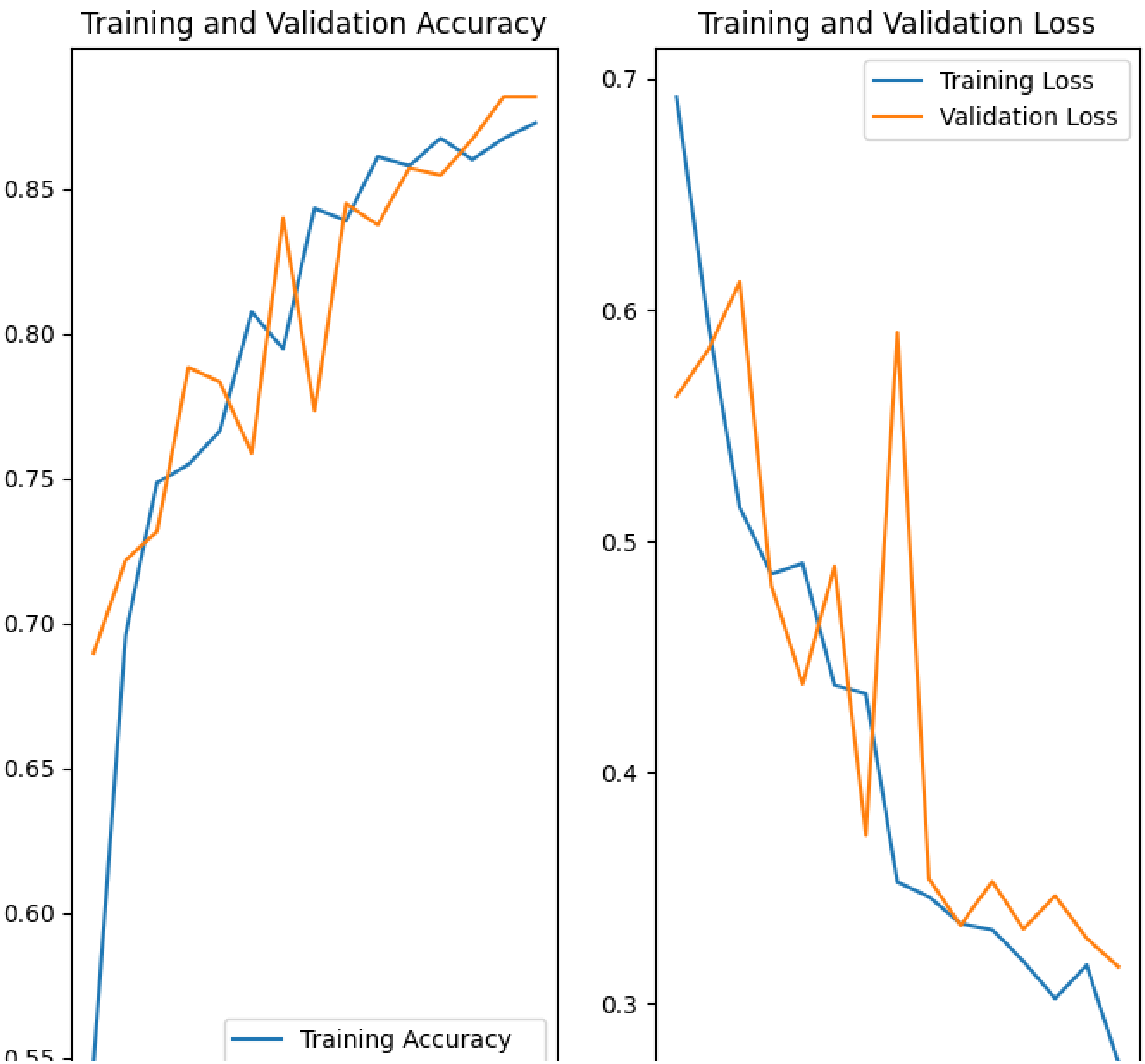
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']


loss = history.history['loss']
val_loss = history.history['val_loss']


epochs_range = range(epochs)


plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')


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


Training and Validation Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
Training Accuracy
Training and Validation Loss
0.7
0.6
0.5
0.4
0.3
Training Loss
Validation Loss

```
#menampilkan hasil

from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
    temp_name = fn
    img = tf.keras.utils.load_img(temp_name, target_size=(img_height, img_width))
    plt.imshow(img)

    img_array = tf.keras.utils.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array, batch_size)
    score = tf.nn.sigmoid(predictions[0])

    class_names = ['Bukan Mobil', 'Mobil']

    print(
        "Gambar dibawah merupakan kategori {} dengan akurasi {:.2f}%."
        .format(class_names[int(round(score.numpy()[0] - 0.2))], 100 * score.numpy()[0])
    )
```

Choose Files Screenshot… 203540.png

- **Screenshot 2023-11-28 203540.png**(image/png) - 578865 bytes, last modified: 11/28/2023 - 100% done

```
Saving Screenshot 2023-11-28 203540.png to Screenshot 2023-11-28 203540.png
1/1 [==============================] - 0s 315ms/step
Gambar dibawah merupakan kategori Mobil dengan akurasi 73.10%.
```