TEAM ID - PNT2022TMID44460

Import and unzip the dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#unzip the downloaded dataset
!unzip '/content/drive/MyDrive/damage  vehicle.zip'
```

```
Archive: /content/drive/MyDrive/damage
   vehicle.zipcreating: damage
   vehicle/
   creating:  damage
   vehicle/body/
   creating:  damage
   vehicle/body/trainin
   g/
   creating:  damage
  vehicle/body/training/00-front/
  inflating: damage
  vehicle/body/training/00-front/0001.jpeg
  inflating: damage
  vehicle/body/training/00-front/0002.JPEG
  inflating: damage
  vehicle/body/training/00-front/0003.JPEG
  inflating: damage
  vehicle/body/training/00-front/0004.JPEG
  inflating: damage
  vehicle/body/training/00-front/0005.JPEG
  inflating: damage
  vehicle/body/training/00-front/0006.JPEG
  inflating: damage
  vehicle/body/training/00-front/0007.JPEG
  inflating: damage
  vehicle/body/training/00-front/0008.jpeg
  inflating: damage
  vehicle/body/training/00-front/0009.JPEG
  inflating: damage
  vehicle/body/training/00-front/0010.JPEG
  inflating: damage
  vehicle/body/training/00-front/0011.JPEG
  inflating: damage
  vehicle/body/training/00-front/0012.jpeg
  inflating: damage
  vehicle/body/training/00-front/0013.JPEG
  inflating: damage
  vehicle/body/training/00-front/0014.JPEG
  inflating: damage
  vehicle/body/training/00-front/0015.JPEG
  inflating: damage
  vehicle/body/training/00-front/0016.JPEG
  inflating: damage
  vehicle/body/training/00-front/0017.JPEG
  inflating: damage
  vehicle/body/training/00-front/0018.JPEG
  inflating: damage
  vehicle/body/training/00-front/0019.JPEG
  inflating: damage
```

```
vehicle/body/training/00-front/0020.jpeg
inflating: damage
vehicle/body/training/00-front/0021.JPEG
inflating: damage
vehicle/body/training/00-front/0022.JPEG
inflating: damage
vehicle/body/training/00-front/0023.JPEG
inflating: damage
vehicle/body/training/00-front/0024.JPEG
inflating: damage
vehicle/body/training/00-front/0025.jpeg
inflating: damage
vehicle/body/training/00-front/0026.JPEG
inflating: damage
vehicle/body/training/00-front/0027.JPEG
inflating: damage
vehicle/body/training/00-front/0028.JPEG
inflating: damage
vehicle/body/training/00-front/0029.JPEG
inflating: damage
vehicle/body/training/00-front/0030.JPEG
inflating: damage
vehicle/body/training/00-front/0031.JPEG
inflating: damage
vehicle/body/training/00-front/0032.JPEG
inflating: damage
vehicle/body/training/00-front/0033.JPEG
inflating: damage
vehicle/body/training/00-front/0034.JPEG
inflating: damage
vehicle/body/training/00-front/0035.jpeg
inflating: damage
vehicle/body/training/00-front/0036.JPEG
inflating: damage
vehicle/body/training/00-front/0037.JPEG
inflating: damage
vehicle/body/training/00-front/0038.JPEG
inflating: damage
vehicle/body/training/00-front/0039.JPEG
inflating: damage
vehicle/body/training/00-front/0040.JPEG
inflating: damage
vehicle/body/training/00-front/0041.JPEG
inflating: damage
vehicle/body/training/00-front/0042.JPEG
inflating: damage
vehicle/body/training/00-front/0043.JPEG
inflating: damage
vehicle/body/training/00-front/0044.JPEG
inflating: damage
vehicle/body/training/00-front/0045.JPEG
inflating: damage
vehicle/body/training/00-front/0046.jpeg
inflating: damage
vehicle/body/training/00-front/0047.JPEG
inflating: damage
vehicle/body/training/00-front/0048.JPEG
inflating: damage
vehicle/body/training/00-front/0049.JPEG
inflating: damage
vehicle/body/training/00-front/0050.JPEG
inflating: damage
vehicle/body/training/00-front/0051.JPEG
inflating: damage
vehicle/body/training/00-front/0052.JPEG
inflating:  damage
vehicle/body/training/00-front/0053.JPEG
```

Image Preprocessing

- **Import The ImageDataGenerator Library**

```
from  tensorflow.keras.preprocessing.image  import  ImageDataGenerator
```

- **Configure ImageDataGenerator Class**

```
#Creating augmentation on training
variable train_datagen  =
ImageDataGenerator(rescale=1./255,
                                    shear_range =
                                    0.1,
                                    zoom_range=0.1,
                                    horizontal_flip=T
                                    rue)

# Creating augmentation on testing variable

test_datagen  =  ImageDataGenerator(rescale=1./255)
```

- **Apply ImageDataGenerator Functionality To Trainset And Testset**

```
# Passing training data to train variable for body

xtrain  =  train_datagen.flow_from_directory('/content/damage  vehicle/body/training',
                                             target_size=(224,22
                                             4),
                                             class_mode='categor
                                             ical',
                                             batch_size=10)


      Found 979 images belonging to 3 classes.

# Passing testing data to test variable for body

xtest  =  test_datagen.flow_from_directory('/content/damage  vehicle/body/validation',
                                           target_size
                                           =(224,224),
                                           class_mode=
                                           'categorica
                                           l',
                                           batch_size=
                                           10)


      Found 171 images belonging to 3 classes.

# Passing training data to train variable for level

x_train  =  train_datagen.flow_from_directory('/content/damage  vehicle/level/training',
                                             target_size=(224,22
                                             4),
                                             class_mode='categor
                                             ical',
                                             batch_size=10)


      Found 979 images belonging to 3 classes.

# Passing testing data to test variable for level

x_test  =  test_datagen.flow_from_directory('/content/damage  vehicle/level/validation',
                                           target_size
                                           =(224,224),
```

```
                                        class_mode=
                                        'categorica
                                        l',
                                        batch_size=
                                        10)
```

```
Found 171 images belonging to 3 classes.
```

Model Building

For Body

- **Importing The Model Building Libraries**

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

import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda,
Dense, Flatten from tensorflow.keras.models import
Model
from tensorflow.keras.applications.vgg16
import            VGG16            from
tensorflow.keras.applications.vgg19
import            VGG19            from
tensorflow.keras.preprocessing    import
image
from tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img from tensorflow.keras.models import
Sequential
import
numpy
as np
from
glob
import
glob
```

- **Loading The Model**

```
IMAGE_SIZE = [224, 224]

train_path = '/content/damage
vehicle/body/training' valid_path =
'/content/damage
vehicle/body/validation'

vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_ke 58889256/58889256 [==============================] - 3s
0us/step
```

- **Adding Flatten Layer**

```
for layer in
    vgg16.la
    yers:
    layer.tr
    ainable
    = False

folders = glob('/content/damage  vehicle/body/training/*')
```

```
['/content/damage
 vehicle/body/training/00-front',
 '/content/damage
 vehicle/body/training/01-rear',
 '/content/damage
 vehicle/body/training/02-side']
```

```
x = Flatten()(vgg16.output)

len(folders)
```

```
3
```

- **Adding Output Layer**

```
prediction = Dense(len(folders), activation='softmax')(x)
```

- **Creating A Model Object**

```
model = Model(inputs=vgg16.input, outputs=prediction)

model.summary()
```

```
Model: "model"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 3)	75267

```
=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
```

- **Configure The Learning Process**

```
model.compile(
  loss='categorical_cro
  ssentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

- **Train The Model**

```
r =
  model.fit_gen
  erator(
  xtrain,
  validation_da
  ta=xtest,
  epochs=25,
  steps_per_epo
  ch=len(xtrain
  ),
  validation_st
  eps=len(xtest
  )
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_generator` is
deprecated and will be
```

Epoch	1/25										
98/98	[==============================]	-	23s	146ms/step	- loss:	1.2077	- accuracy:	0.5465	- val_loss:	1.2900	- val_accuracy:
Epoch	2/25										
98/98	[==============================]	-	13s	128ms/step	• loss:	0.8364	• accuracy:	0.7028	• val_loss:	0.8665	• val_accuracy:
Epoch	3/25										
98/98	[==============================]	-	13s	128ms/step	● loss:	0.5293	● accuracy:	0.7998	● val_loss:	1.3260	● val_accuracy:
Epoch	4/25										
98/98	[==============================]	-	12s	127ms/step	- loss:	0.3978	- accuracy:	0.8611	- val_loss:	0.9842	- val_accuracy:
Epoch	5/25										
98/98	[==============================]	-	12s	127ms/step	• loss:	0.2783	• accuracy:	0.9030	• val_loss:	0.9397	• val_accuracy:
Epoch	6/25										
98/98	[==============================]	-	13s	128ms/step	● loss:	0.2690	● accuracy:	0.9070	● val_loss:	0.9892	● val_accuracy:
Epoch	7/25										
98/98	[==============================]	-	12s	127ms/step	- loss:	0.1788	- accuracy:	0.9448	- val_loss:	1.0052	- val_accuracy:
Epoch	8/25										
98/98	[==============================]	-	13s	129ms/step	- loss:	0.1671	- accuracy:	0.9469	- val_loss:	1.1693	- val_accuracy:
Epoch	9/25										
98/98	[==============================]	-	13s	129ms/step	• loss:	0.1277	• accuracy:	0.9561	• val_loss:	1.0058	• val_accuracy:
Epoch	10/25										
98/98	[==============================]	-	13s	128ms/step	● loss:	0.1184	● accuracy:	0.9591	● val_loss:	1.0620	● val_accuracy:
Epoch	11/25										

98/98	[==============================]	-	13s	130ms/step	- loss:	0.0963	- accuracy:	0.9745	- val_loss:	1.1219	- val_accuracy:
Epoch	12/25										

98/98 Epoch	[==============================] 13/25	- 13s	129ms/step	- loss:	0.0857	- accuracy:	0.9765	- val_loss:	1.0284	- val_accuracy:
98/98 Epoch 98/98	[==============================] 14/25 [==============================]	- 13s - 13s	129ms/step 129ms/step	• loss: • loss:	0.0582 0.0688	• accuracy: • accuracy:	0.9837 0.9877	• val_loss: • val_loss:	1.1153 1.1033	• val_accuracy: • val_accuracy:
Epoch 98/98	15/25 [==============================]	- 13s	131ms/step	- loss:	0.0709	- accuracy:	0.9867	- val_loss:	1.0730	- val_accuracy:
Epoch 98/98	16/25 [==============================]	- 13s	128ms/step	- loss:	0.0895	- accuracy:	0.9775	- val_loss:	1.1225	- val_accuracy:
Epoch	17/25									
98/98 Epoch 98/98	[==============================] 18/25 [==============================]	- 13s - 13s	129ms/step 128ms/step	• loss: • loss:	0.0609 0.0998	• accuracy: • accuracy:	0.9918 0.9714	• val_loss: • val_loss:	1.2937 1.1754	• val_accuracy: • val_accuracy:
Epoch 98/98	19/25 [==============================]	- 13s	128ms/step	- loss:	0.0728	- accuracy:	0.9847	- val_loss:	1.5074	- val_accuracy:
Epoch	20/25									
98/98 Epoch 98/98	[==============================] 21/25 [==============================]	- 13s - 13s	129ms/step 131ms/step	• loss: • loss:	0.0972 0.0404	• accuracy: • accuracy:	0.9714 0.9908	• val_loss: • val_loss:	1.4684 1.4215	• val_accuracy: • val_accuracy:
Epoch 98/98	22/25 [==============================]	- 13s	131ms/step	- loss:	0.0854	- accuracy:	0.9867	- val_loss:	1.4772	- val_accuracy:
Epoch 98/98	23/25 [==============================]	- 13s	128ms/step	- loss:	0.0399	- accuracy:	0.9918	- val_loss:	1.4306	- val_accuracy:
Epoch	24/25									
98/98 Epoch 98/98	[==============================] 25/25 [==============================]	- 13s - 13s	129ms/step 129ms/step	• loss: • loss:	0.0400 0.1692	• accuracy: • accuracy:	0.9908 0.9387	• val_loss: • val_loss:	1.4562 1.6805	• val_accuracy: • val_accuracy:

- **Save The Model**

```
from tensorflow.keras.models import load_model

model.save('/content/damage  vehicle/Model/body.h5')
```

- **Test The Model**

```
from tensorflow.keras.models
```

```
import load_modelimport cv2
from skimage.transform import resize
```

```
model = load_model('/content/damage vehicle/Model/body.h5')
```

```
def detect(frame):
  img = cv2.resize(frame,(224,224))
  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

  i
  f
  (
  n
  p
  .
  m
  a
  x
  (
  i
  m
  g
  )
  >
  1
  )
  :
  i
  m
  g
  =
  i
  m
  g
  /
  2
  5
  5
  .
  0
  img =
  np.array([img]
  ) prediction
  =
  model.predict(
  img)label =
  ["front","rear
  ","side"]
  preds = label[np.argmax(prediction)]
  return preds
```

```
import numpy as np
```

```
data = "/content/damage vehicle/body/training/00
-front/0002.JPEG" image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 0s 148ms/step
front
```

M

o

d

el

B

ui

ld

in

g

F

o

r

L

e

v

el

- **Importing The Model Building Libraries**

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda,
Dense, Flatten from tensorflow.keras.models import
Model
from tensorflow.keras.applications.vgg16
import VGG16 from
tensorflow.keras.applications.vgg19
import VGG19 from
```

```
tensorflow.keras.preprocessing    import
image
from tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img from tensorflow.keras.models import
Sequential
import
numpy
as np
from
glob
import
glob
```

- **Loading The Model**

```
IMAGE_SIZE = [224, 224]

train_path = '/content/damage
vehicle/level/training' valid_path  =
'/content/damage  vehicle/level/validation'

vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

- **Adding Flatten Layer**

```
for layer in
    vgg16.la
    yers:
    layer.tr
    ainable
    = False

folders = glob('/content/damage  vehicle/level/training/*')
```

```
['/content/damage
 vehicle/level/training/03-severe',
 '/content/damage
 vehicle/level/training/02-moderate',
 '/content/damage
 vehicle/level/training/01-minor']
```

```
x = Flatten()(vgg16.output)

len(folders)
```

```
3
```

- **Adding Output Layer**

```
prediction = Dense(len(folders), activation='softmax')(x)
```

- **Creating A Model Object**

```
model = Model(inputs=vgg16.input, outputs=prediction)

model.summary()
```

```
Model: "model_1"
```

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0

block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 3)	75267

```
=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
```

- **Configure The Learning Process**

```
model.compile(
  loss='categorical_cro
  ssentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

- **Train The Model**

```
r =
  model.fit_gener
  ator( x_train,
  validation_data
  =x_test,
  epochs=25,
  steps_per_epoch
  =len(x_train),
  validation_step
  s=len(x_test)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_generator` is
deprecated and will be
```

Epoch	1/25									
98/98 [==========] 2/25	[==========]	-	14s	133ms/step	• loss:	1.1629	• accuracy:	0.5495	• val_loss:	1.1559	• val_accuracy:
Epoch 98/98	[==========]	-	13s	130ms/step	• loss:	0.7157	• accuracy	0.7089	• val_loss	0.9643	• val_accuracy

							:		:		:
Epoch 98/98	3/25 [==========]	-	13s	130ms/step	- loss:	0.4978	- accuracy:	0.8161	- val_loss:	1.5663	- val_accuracy:
Epoch 98/98	4/25 [==========]	-	13s	128ms/step	- loss:	0.5277	- accuracy:	0.7865	- val_loss:	1.6003	- val_accuracy:
Epoch	5/25										
98/98 Epoch 98/98	[==========] 6/25 [==========]	-	13s 13s	128ms/step 128ms/step	• loss: ● loss:	0.3763 0.2445	• accuracy: ● accuracy:	0.8468 0.9203	• val_loss: ● val_loss:	1.1925 1.0354	• val_accuracy: ● val_accuracy:
Epoch 98/98	7/25 [==========]	-	13s	128ms/step	- loss:	0.1902	- accuracy:	0.9346	- val_loss:	1.2155	- val_accuracy:
Epoch	8/25										
98/98 Epoch	[==========] 9/25	-	13s	128ms/step	- loss:	0.1327	- accuracy:	0.9571	- val_loss:	1.0902	- val_accuracy:
98/98 Epoch 98/98	[==========] 10/25 [==========]	-	13s 13s	127ms/step 128ms/step	• loss: ● loss:	0.1206 0.1181	• accuracy: ● accuracy:	0.9540 0.9591	• val_loss: ● val_loss:	1.1282 1.1311	• val_accuracy: ● val_accuracy:
Epoch 98/98	11/25 [==========]	-	13s	128ms/step	- loss:	0.0910	- accuracy:	0.9765	- val_loss:	1.1538	- val_accuracy:
Epoch	12/25										
98/98 Epoch 98/98	[==========] 13/25 [==========]	-	12s 13s	127ms/step 128ms/step	• loss: ● loss:	0.0813 0.0603	• accuracy: ● accuracy:	0.9806 0.9857	• val_loss: ● val_loss:	1.2209 1.2545	• val_accuracy: ● val_accuracy:
Epoch	14/25										

98/98 Epoch	[==========] 15/25	-	12s	127ms/step	- loss:	0.0474	- accuracy:	0.9949	- val_loss:	1.1609	- val_accuracy:
98/98 Epoch 98/98	[==========] 16/25 [==========]	-	13s 13s	129ms/step 128ms/step	• loss: ● loss:	0.0366 0.0493	• accuracy: ● accuracy:	0.9959 0.9888	• val_loss: ● val_loss:	1.1688 1.1850	• val_accuracy: ● val_accuracy:
Epoch 98/98	17/25 [==========]	-	13s	128ms/step	- loss:	0.0320	- accuracy:	0.9939	- val_loss:	1.1884	- val_accuracy:
Epoch 98/98	18/25 [==========]	-	13s	129ms/step	- loss:	0.0363	- accuracy:	0.9939	- val_loss:	1.2897	- val_accuracy:
Epoch	19/25										
98/98 Epoch 98/98	[==========] 20/25 [==========]	-	13s 13s	128ms/step 130ms/step	• loss: ● loss:	0.0298 0.0250	• accuracy: ● accuracy:	0.9949 0.9980	• val_loss: ● val_loss:	1.2499 1.2801	• val_accuracy: ● val_accuracy:
Epoch 98/98	21/25 [==========]	-	13s	129ms/step	- loss:	0.0329	- accuracy:	0.9959	- val_loss:	1.2366	- val_accuracy:
Epoch	22/25										
98/98 Epoch 98/98	[==========] 23/25 [==========]	-	13s 13s	128ms/step 130ms/step	• loss: ● loss:	0.0170 0.0216	• accuracy: ● accuracy:	1.0000 1.0000	• val_loss: ● val_loss:	1.2901 1.2697	• val_accuracy: ● val_accuracy:

Epoch 98/98	24/25 [==============================]	-	13s	128ms/step	- loss:	0.0365	- accuracy:	0.9908	- val_loss:	1.4214	- val_accuracy:
Epoch 98/98	25/25 [==============================]	-	13s	129ms/step	- loss:	0.0380	- accuracy:	0.9939	- val_loss:	1.4219	- val_accuracy:

- **Save The Model**

```
from tensorflow.keras.models import load_model

model.save('/content/damage  vehicle/Model/level.h5')
```

- **Test The Model**

```
from tensorflow.keras.models
import load_modelimport cv2
from skimage.transform import resize
```

```
model = load_model('/content/damage  vehicle/Model/level.h5')
```

```
def detect(frame):
  img = cv2.resize(frame,(224,224))
  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

  i
  f
  (
  n
  p
  .
  m
  a
  x
  (
  i
  m
  g
  )
  >
  1
  )
  :
  i
  m
  g
  =
  i
  m
  g
  /
  2
  5
  5
  .
  0
  img =
  np.array([img]
  ) prediction =
  model.predict(
  img)
  label = ["minor","moderate","severe"]
  preds =
  label[np.argmax(predictio
```

```
  n)]return preds
```

```

import numpy as np
```

```

data = "/content/damage  vehicle/level/validation/01
-minor/0005.JPEG"image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 0s 142ms/step
minor
```