## FOR BODY DAMAGE

## IMAGE PRE PROCESSING

### 1. Import The ImageDataGenerator Library

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

### :2. Configure ImageDataGenerator Class

### Image Data Augmentation

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.1,
                                   zoom_range = 0.1,
                                   horizontal_flip = True)

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

### 3. Apply ImageDataGenerator Functionality To Trainset And Testset

```
training_set = 
train_datagen.flow_from_directory('/content/drive/MyDrive/IBM -
PROJECT/Data set/body-20221023T072112Z-001/body/training',
                                                 target_size = (224,
224),
                                                 batch_size = 10,
                                                 class_mode =
'categorical')
test_set = 
test_datagen.flow_from_directory('/content/drive/MyDrive/IBM -
PROJECT/Data set/body-20221023T072112Z-001/body/validation',
                                            target_size = (224, 224),
                                            batch_size = 10,
                                            class_mode =
'categorical')
```

```
Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.
```

## MODEL BUILDING

### 1. 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
```

## 2. Loading The Model

```
IMAGE_SIZE = [224, 224]

train_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training'
valid_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/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_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

## 3. Adding Flatten Layer

```
for layer in vgg16.layers:
    layer.trainable = False

folders = glob('/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/*')

folders
```

```
['/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/02-side',
 '/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/01-rear',
 '/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/00-front']
```

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

len(folders)
```

```
3
```

**4. Adding Output Layer**

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

**5. 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
_________________________________________________________________
```

**6. Configure The Learning Process**

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

**7. Train The Model**

```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=25,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.


Epoch 1/25
98/98 [==============================] - 560s 6s/step - loss: 1.2275 -
accuracy: 0.5383 - val_loss: 0.8698 - val_accuracy: 0.6608
Epoch 2/25
98/98 [==============================] - 584s 6s/step - loss: 0.7810 -
accuracy: 0.7007 - val_loss: 0.8931 - val_accuracy: 0.6491
Epoch 3/25
98/98 [==============================] - 538s 5s/step - loss: 0.4842 -
accuracy: 0.8264 - val_loss: 0.8348 - val_accuracy: 0.6842
Epoch 4/25
98/98 [==============================] - 537s 5s/step - loss: 0.3813 -
accuracy: 0.8560 - val_loss: 0.9010 - val_accuracy: 0.6901
Epoch 5/25
98/98 [==============================] - 537s 5s/step - loss: 0.2735 -
accuracy: 0.8999 - val_loss: 1.0660 - val_accuracy: 0.6901
Epoch 6/25
98/98 [==============================] - 538s 5s/step - loss: 0.2211 -
accuracy: 0.9295 - val_loss: 1.0073 - val_accuracy: 0.7076
Epoch 7/25
98/98 [==============================] - 536s 5s/step - loss: 0.2163 -
accuracy: 0.9224 - val_loss: 0.9560 - val_accuracy: 0.7251
Epoch 8/25
98/98 [==============================] - 538s 6s/step - loss: 0.1728 -
accuracy: 0.9397 - val_loss: 1.0719 - val_accuracy: 0.6491
Epoch 9/25
98/98 [==============================] - 540s 6s/step - loss: 0.1423 -
accuracy: 0.9581 - val_loss: 1.0706 - val_accuracy: 0.6901
```

```
Epoch 10/25
98/98 [==============================] - 539s 6s/step - loss: 0.1118 -
accuracy: 0.9704 - val_loss: 1.1651 - val_accuracy: 0.6842
Epoch 11/25
98/98 [==============================] - 538s 5s/step - loss: 0.0808 -
accuracy: 0.9785 - val_loss: 1.1212 - val_accuracy: 0.7076
Epoch 12/25
98/98 [==============================] - 549s 6s/step - loss: 0.0751 -
accuracy: 0.9857 - val_loss: 1.1451 - val_accuracy: 0.6842
Epoch 13/25
98/98 [==============================] - 555s 6s/step - loss: 0.0730 -
accuracy: 0.9816 - val_loss: 1.0812 - val_accuracy: 0.6842
Epoch 14/25
98/98 [==============================] - 535s 5s/step - loss: 0.1074 -
accuracy: 0.9734 - val_loss: 1.2204 - val_accuracy: 0.6842
Epoch 15/25
98/98 [==============================] - 539s 6s/step - loss: 0.0598 -
accuracy: 0.9888 - val_loss: 1.6480 - val_accuracy: 0.6316
Epoch 16/25
98/98 [==============================] - 543s 6s/step - loss: 0.0810 -
accuracy: 0.9806 - val_loss: 1.2050 - val_accuracy: 0.6901
Epoch 17/25
98/98 [==============================] - 541s 6s/step - loss: 0.1196 -
accuracy: 0.9632 - val_loss: 1.3478 - val_accuracy: 0.6374
Epoch 18/25
98/98 [==============================] - 543s 6s/step - loss: 0.0915 -
accuracy: 0.9755 - val_loss: 1.2961 - val_accuracy: 0.7018
Epoch 19/25
98/98 [==============================] - 544s 6s/step - loss: 0.0687 -
accuracy: 0.9806 - val_loss: 1.2175 - val_accuracy: 0.6842
Epoch 20/25
98/98 [==============================] - 546s 6s/step - loss: 0.0492 -
accuracy: 0.9918 - val_loss: 1.3791 - val_accuracy: 0.6784
Epoch 21/25
98/98 [==============================] - 543s 6s/step - loss: 0.0674 -
accuracy: 0.9847 - val_loss: 1.5585 - val_accuracy: 0.6433
Epoch 22/25
98/98 [==============================] - 537s 5s/step - loss: 0.0740 -
accuracy: 0.9775 - val_loss: 1.7693 - val_accuracy: 0.6550
Epoch 23/25
98/98 [==============================] - 538s 6s/step - loss: 0.0822 -
accuracy: 0.9765 - val_loss: 1.9127 - val_accuracy: 0.6374
Epoch 24/25
98/98 [==============================] - 541s 6s/step - loss: 0.1048 -
accuracy: 0.9653 - val_loss: 1.5448 - val_accuracy: 0.6316
Epoch 25/25
98/98 [==============================] - 544s 6s/step - loss: 0.1373 -
accuracy: 0.9551 - val_loss: 1.4574 - val_accuracy: 0.6842
```

**8. Save The Model**

```
from tensorflow.keras.models import load_model

model.save('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Model/body.h5')
```

## 9. Test The Model

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

model = load_model('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Model/body.h5')

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

  if(np.max(img)>1):
    img = img/255.0
  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/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/00-front/0008.jpeg"
image = cv2.imread(data)
print(detect(image))

1/1 [==============================] - 0s 498ms/step
front
```

FOR LEVEL DAMAGE

IMAGE PRE PROCESSING

1. Import The ImageDataGenerator Library

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

1. Configure ImageDataGenerator Class

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.1,
                                   zoom_range = 0.1,
                                   horizontal_flip = True)

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

1. Apply ImageDataGenerator Functionality To Trainset And Testset

```
training_set = 
train_datagen.flow_from_directory('/content/drive/MyDrive/IBM - 
PROJECT/Data set/level-20221023T072121Z-001/level/training',
                                                 target_size = (224, 
224),
                                                 batch_size = 10,
                                                 class_mode = 
'categorical')
test_set = 
test_datagen.flow_from_directory('/content/drive/MyDrive/IBM - 
PROJECT/Data set/level-20221023T072121Z-001/level/validation',
                                            target_size = (224, 224),
                                            batch_size = 10,
                                            class_mode = 
'categorical')
```

```
Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.
```

## MODEL BUILDING

### 1. 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
```

## 2. Loading The Model

```
IMAGE_SIZE = [224, 224]

train_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/level-
20221023T072121Z-001/level/training'
valid_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/level-
20221023T072121Z-001/level/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_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

### 3. Adding Flatten Layer

```
for layer in vgg16.layers:
    layer.trainable = False

folders = glob('/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/*')

folders

['/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/03-severe',
 '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/02-moderate',
 '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/01-minor']

x = Flatten()(vgg16.output)

len(folders)

3
```

### 4. Adding Output Layer

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

### 5. 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
_________________________________________________________________
```

### 6. Configure The Learning Process

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

### 7. Train The Model

```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=25,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
```

generators.

```
Epoch 1/25
98/98 [==============================] - 606s 6s/step - loss: 1.1697 -
accuracy: 0.5608 - val_loss: 0.9855 - val_accuracy: 0.6140
Epoch 2/25
98/98 [==============================] - 596s 6s/step - loss: 0.7030 -
accuracy: 0.7099 - val_loss: 0.9670 - val_accuracy: 0.6199
Epoch 3/25
98/98 [==============================] - 594s 6s/step - loss: 0.4431 -
accuracy: 0.8202 - val_loss: 1.0758 - val_accuracy: 0.5965
Epoch 4/25
98/98 [==============================] - 592s 6s/step - loss: 0.3887 -
accuracy: 0.8570 - val_loss: 1.0519 - val_accuracy: 0.6257
Epoch 5/25
98/98 [==============================] - 592s 6s/step - loss: 0.3058 -
accuracy: 0.8856 - val_loss: 1.5903 - val_accuracy: 0.6140
Epoch 6/25
98/98 [==============================] - 596s 6s/step - loss: 0.2978 -
accuracy: 0.9019 - val_loss: 1.1763 - val_accuracy: 0.6140
Epoch 7/25
98/98 [==============================] - 598s 6s/step - loss: 0.2060 -
accuracy: 0.9295 - val_loss: 1.2846 - val_accuracy: 0.6082
Epoch 8/25
98/98 [==============================] - 596s 6s/step - loss: 0.1685 -
accuracy: 0.9387 - val_loss: 1.1337 - val_accuracy: 0.6023
Epoch 9/25
98/98 [==============================] - 595s 6s/step - loss: 0.1926 -
accuracy: 0.9305 - val_loss: 1.1559 - val_accuracy: 0.6725
Epoch 10/25
98/98 [==============================] - 594s 6s/step - loss: 0.1206 -
accuracy: 0.9653 - val_loss: 1.2013 - val_accuracy: 0.6433
Epoch 11/25
98/98 [==============================] - 595s 6s/step - loss: 0.1151 -
accuracy: 0.9663 - val_loss: 1.2582 - val_accuracy: 0.6023
Epoch 12/25
98/98 [==============================] - 595s 6s/step - loss: 0.0615 -
accuracy: 0.9857 - val_loss: 1.1696 - val_accuracy: 0.6608
Epoch 13/25
98/98 [==============================] - 597s 6s/step - loss: 0.0659 -
accuracy: 0.9837 - val_loss: 1.1735 - val_accuracy: 0.6374
Epoch 14/25
98/98 [==============================] - 597s 6s/step - loss: 0.0417 -
accuracy: 0.9939 - val_loss: 1.1479 - val_accuracy: 0.6433
Epoch 15/25
98/98 [==============================] - 597s 6s/step - loss: 0.0504 -
accuracy: 0.9898 - val_loss: 1.5237 - val_accuracy: 0.5673
Epoch 16/25
98/98 [==============================] - 596s 6s/step - loss: 0.0437 -
```

```
accuracy: 0.9888 - val_loss: 1.4307 - val_accuracy: 0.6140
Epoch 17/25
98/98 [==============================] - 602s 6s/step - loss: 0.0428 -
accuracy: 0.9877 - val_loss: 1.2403 - val_accuracy: 0.6433
Epoch 18/25
98/98 [==============================] - 605s 6s/step - loss: 0.0359 -
accuracy: 0.9949 - val_loss: 1.3156 - val_accuracy: 0.6433
Epoch 19/25
98/98 [==============================] - 598s 6s/step - loss: 0.0289 -
accuracy: 0.9959 - val_loss: 1.4142 - val_accuracy: 0.6140
Epoch 20/25
98/98 [==============================] - 594s 6s/step - loss: 0.0256 -
accuracy: 0.9980 - val_loss: 1.3567 - val_accuracy: 0.6316
Epoch 21/25
98/98 [==============================] - 598s 6s/step - loss: 0.0248 -
accuracy: 0.9990 - val_loss: 1.3492 - val_accuracy: 0.6257
Epoch 22/25
98/98 [==============================] - 596s 6s/step - loss: 0.0222 -
accuracy: 1.0000 - val_loss: 1.3326 - val_accuracy: 0.6491
Epoch 23/25
98/98 [==============================] - 597s 6s/step - loss: 0.0137 -
accuracy: 0.9990 - val_loss: 1.4157 - val_accuracy: 0.6199
Epoch 24/25
98/98 [==============================] - 595s 6s/step - loss: 0.0398 -
accuracy: 0.9888 - val_loss: 1.4562 - val_accuracy: 0.6257
Epoch 25/25
98/98 [==============================] - 597s 6s/step - loss: 0.0292 -
accuracy: 0.9939 - val_loss: 1.5857 - val_accuracy: 0.5965
```

## 8. Save The Model

```
from tensorflow.keras.models import load_model

model.save('/content/drive/MyDrive/Intelligent Vehicle Damage
Assessment & Cost Estimator For Insurance Companies/Model/level.h5')
```

## 9. Test The Model

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


model = load_model('/content/drive/MyDrive/Intelligent Vehicle Damage
Assessment & Cost Estimator For Insurance Companies/Model/level.h5')

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

  if(np.max(img)>1):
    img = img/255.0
```

```
  img = np.array([img])
  prediction = model.predict(img)
  label = ["minor","moderate","severe"]
  preds = label[np.argmax(prediction)]
  return preds
```

```
import numpy as np
```

```
data = "/content/drive/MyDrive/IBM - PROJECT/Data set/level-
20221023T072121Z-001/level/validation/01-minor/0008.jpeg"
image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 1s 674ms/step
minor
```