Unzip data

```
!unzip '/content/Car Damage.zip'
```

```
      inflating: Car damage/body/training/01-rear/0042.JPEG  
      inflating: Car damage/level/training/03-severe/0185.JPEG  
      inflating: Car damage/body/training/01-rear/0043.JPEG  
      inflating: Car damage/body/training/01-rear/0038.JPEG  
      inflating: Car damage/level/training/03-severe/0187.JPEG  
      inflating: Car damage/level/training/03-severe/0201.JPEG  
      inflating: Car damage/level/training/03-severe/0251.JPEG  
      inflating: Car damage/level/training/03-severe/0190.JPEG  
      inflating: Car damage/level/training/03-severe/0189.JPEG  
      inflating: Car damage/body/training/01-rear/0211.jpeg  
      inflating: Car damage/level/training/03-severe/0219.JPEG  
      inflating: Car damage/level/training/03-severe/0183.JPEG  
      inflating: Car damage/level/training/03-severe/0295.JPEG  
      inflating: Car damage/body/training/01-rear/0037.JPEG  
      inflating: Car damage/level/training/03-severe/0181.JPEG  
      inflating: Car damage/body/training/01-rear/0051.JPEG  
      inflating: Car damage/level/training/03-severe/0228.JPEG  
      inflating: Car damage/level/training/03-severe/0222.JPEG  
      inflating: Car damage/level/training/03-severe/0272.JPEG  
      inflating: Car damage/level/training/03-severe/0191.JPEG  
      inflating: Car damage/level/training/03-severe/0292.JPEG  
      inflating: Car damage/level/training/03-severe/0282.JPEG  
      inflating: Car damage/level/training/03-severe/0180.JPEG  
      inflating: Car damage/level/training/03-severe/0186.jpeg  
      inflating: Car damage/level/training/03-severe/0245.JPEG  
      inflating: Car damage/level/training/03-severe/0269.JPEG  
      inflating: Car damage/level/training/03-severe/0293.JPEG  
      inflating: Car damage/level/training/03-severe/0200.jpeg  
      inflating: Car damage/level/training/03-severe/0298.JPEG  
      inflating: Car damage/level/training/03-severe/0203.JPEG  
      inflating: Car damage/level/training/03-severe/0217.jpeg  
      inflating: Car damage/level/training/03-severe/0188.JPEG  
      inflating: Car damage/level/training/03-severe/0290.JPEG  
      inflating: Car damage/level/training/03-severe/0248.jpeg  
      inflating: Car damage/level/training/03-severe/0202.JPEG  
      inflating: Car damage/level/training/03-severe/0211.JPEG  
      inflating: Car damage/level/training/03-severe/0197.jpeg  
      inflating: Car damage/level/training/03-severe/0247.JPEG  
      inflating: Car damage/level/training/03-severe/0179.jpeg  
      inflating: Car damage/level/training/03-severe/0236.jpeg  
      inflating: Car damage/level/training/03-severe/0017.JPEG  
      inflating: Car damage/level/training/03-severe/0022.JPEG  
      inflating: Car damage/level/training/03-severe/0012.JPEG  
      inflating: Car damage/level/training/03-severe/0297.JPEG  
      inflating: Car damage/level/training/03-severe/0015.JPEG  
      inflating: Car damage/level/training/03-severe/0182.JPEG  
      inflating: Car damage/level/training/03-severe/0013.JPEG  
      inflating: Car damage/level/training/03-severe/0194.jpeg  
```

```
inflating: Car damage/level/training/03-severe/0194.jpeg
inflating: Car damage/level/training/03-severe/0184.JPEG
inflating: Car damage/level/training/03-severe/0007.JPEG
inflating: Car damage/level/training/03-severe/0195.JPEG
inflating: Car damage/level/training/03-severe/0023.jpeg
inflating: Car damage/level/training/03-severe/0010.JPEG
inflating: Car damage/body/training/01-rear/0048.JPEG
inflating: Car damage/level/training/03-severe/0257.JPEG
inflating: Car damage/level/training/03-severe/0196.JPEG
inflating: Car damage/level/training/03-severe/0021.JPEG
inflating: Car damage/level/training/03-severe/0018.JPEG
```

1. Image Augmentation

```
#import lib.
```

```
imageSize = [224, 224]
```

```
#augmentation on car damage
```

```
train_datagen=ImageDataGenerator(rescale=1./255,
                                 zoom_range=0.2,
                                 horizontal_flip=True)
```

```
valid_datagen=ImageDataGenerator(rescale=1./255,
                                 zoom_range=0.2,
                                 horizontal_flip=True)
```

```
x_train= train_datagen.flow_from_directory('/content/Car damage/body/training',
                                           target_size=(224,224),
                                           class_mode='categorical',
                                           batch_size=10)
```

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

```
x_test= valid_datagen.flow_from_directory('/content/Car damage/body/validation',
                                          target_size=(224,224),
                                          class_mode='categorical',
                                          batch_size=10)
```

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

2.Importing the Model Building Libraries

```
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
```

3. Loading The Model

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg1
58889256/58889256 [==============================] - 1s 0us/step
```

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

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

4.Adding Flatten Layer

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

5.Adding output Layer

```
x = Flatten()(vgg.output)
prediction = Dense(3, activation='softmax')(x)
```

6.Creating the Model

```
model = Model(inputs=vgg.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
_________________________________________________________________
```

7.Train the Model

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

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
```

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

```
x_train= train_datagen.flow_from_directory('/content/Car damage/body/training',
                                           target_size=(224,224),
                                           class_mode='categorical',
                                           batch_size=10)
```

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

```
x_test= valid_datagen.flow_from_directory('/content/Car damage/body/validation',
                                          target_size=(224,224),
                                          class_mode='categorical',
                                          batch_size=10)
```

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

9.Fit the Model

```
r = model.fit_generator(
   x_train,
    validation_data=x_test,
  epochs=5,
  steps_per_epoch=500//10,
  validation_steps=171//10)
```

```
Epoch 1/5
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit

50/50 [==============================] - 352s 7s/step - loss: 1.0610 - accuracy: 0.5671
Epoch 2/5
50/50 [==============================] - 356s 7s/step - loss: 0.9911 - accuracy: 0.6232
Epoch 3/5
50/50 [==============================] - 351s 7s/step - loss: 0.7817 - accuracy: 0.7295
```

```
Epoch 4/5
50/50 [==============================] - 347s 7s/step - loss: 0.6761 - accuracy: 0.7680
Epoch 5/5
50/50 [==============================] - 346s 7s/step - loss: 0.5670 - accuracy: 0.7960
```

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

```
#import load_model class for loading h5 file

from tensorflow.keras.models import load_model
```

```
#import image class to process the images

from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input
import numpy as np
```

```
#load one random image from local system

img=image.load_img(r'/content/Car damage/body/training/00-front/0002.JPEG',target_size=(224,2
```

```
#convert image to array format

x=image.img_to_array(img)
import numpy as np
```

```
x=np.expand_dims(x,axis=0)
```

```
img_data=preprocess_input(x)
```

```
img_data.shape
```

```
(1, 224, 224, 3)
```

```
img_data.shape
```

```
(1, 224, 224, 3)
```

```
model.predict(img_data)
```

```
1/1 [==============================] - 1s 799ms/step
array([[9.9700636e-01, 2.9936929e-03, 2.5396329e-09]], dtype=float32)
```

```
output=np.argmax(model.predict(img_data), axis=1)
```

```
output
```

```
1/1 [==============================] - 1s 530ms/step
array([0])
```

```
vgg1 = VGG16(input_shape=imageSize + [3], weights='imagenet',include_top=False)
```

```
for layer in vgg1.layers:
  layer.trainable = False
```

```
x = Flatten()(vgg1.output)
prediction = Dense(3, activation='softmax')(x)
```

```
model1 = Model(inputs=vgg1.input, outputs=prediction)
```

```
model1.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy'])
```

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
x_train= train_datagen.flow_from_directory('/content/Car damage/level/training',
                                           target_size=(224,224),
                                           class_mode='categorical',
                                           batch_size=10)
```

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

```
x_test= valid_datagen.flow_from_directory('/content/Car damage/level/validation',
                                          target_size=(224,224),
                                          class_mode='categorical',
                                          batch_size=10)
```

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

```
r = model1.fit_generator(
    x_train,
    validation_data=x_test,
    epochs=5,
    steps_per_epoch=500//10,
    validation_steps=171//10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit

Epoch 1/5
50/50 [==============================] - 341s 7s/step - loss: 1.2577 - accuracy: 0.5271
Epoch 2/5
50/50 [==============================] - 343s 7s/step - loss: 0.9081 - accuracy: 0.6693
Epoch 3/5
50/50 [==============================] - 347s 7s/step - loss: 0.8177 - accuracy: 0.6960
Epoch 4/5
50/50 [==============================] - 345s 7s/step - loss: 0.7512 - accuracy: 0.7100
Epoch 5/5
50/50 [==============================] - 349s 7s/step - loss: 0.6532 - accuracy: 0.7720
```

9.Save the Model

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

10.Test the Model

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

✓ 0s completed at 10:36 PM ● ✕