FOR BODY DAMAGE

IMAGE PRE PROCESSING

1. Import The ImageDataGenerator Library

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

2. 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)
```

3. Apply ImageDataGenerator Functionality To Trainset And Testset

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/Car_damage/body/t
                                                 target_size=(224, 224),
                                                 batch_size=10,
                                                 class_mode='categorical')
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/Car_damage/body/valida
                                            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/Car_damage/body/training'
valid_path = '/content/drive/MyDrive/Car_damage/body/validation'


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

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

3. Adding Flatten Layer

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


folders = glob('/content/drive/MyDrive/Car_damage/body/training/*')


folders
```

```
['/content/drive/MyDrive/Car_damage/body/training/02-side',
 '/content/drive/MyDrive/Car_damage/body/training/01-rear',
 '/content/drive/MyDrive/Car_damage/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.f

Epoch 1/25
98/98 [==============================] - 498s 5s/step - loss: 1.3552 - accuracy: 0.53
Epoch 2/25
98/98 [==============================] - 421s 4s/step - loss: 0.6244 - accuracy: 0.76
Epoch 3/25
98/98 [==============================] - 422s 4s/step - loss: 0.4846 - accuracy: 0.81
Epoch 4/25
98/98 [==============================] - 420s 4s/step - loss: 0.3649 - accuracy: 0.86
Epoch 5/25
98/98 [==============================] - 417s 4s/step - loss: 0.2899 - accuracy: 0.89
Epoch 6/25
98/98 [==============================] - 418s 4s/step - loss: 0.3088 - accuracy: 0.88
Epoch 7/25
98/98 [==============================] - 420s 4s/step - loss: 0.1704 - accuracy: 0.93
Epoch 8/25
98/98 [==============================] - 423s 4s/step - loss: 0.1787 - accuracy: 0.93
Epoch 9/25
98/98 [==============================] - 421s 4s/step - loss: 0.1622 - accuracy: 0.94
Epoch 10/25
98/98 [==============================] - 417s 4s/step - loss: 0.0985 - accuracy: 0.97
Epoch 11/25
98/98 [==============================] - 423s 4s/step - loss: 0.0962 - accuracy: 0.96
Epoch 12/25
98/98 [==============================] - 426s 4s/step - loss: 0.0909 - accuracy: 0.98
Epoch 13/25
98/98 [==============================] - 422s 4s/step - loss: 0.0796 - accuracy: 0.98
Epoch 14/25
98/98 [==============================] - 421s 4s/step - loss: 0.0625 - accuracy: 0.98
Epoch 15/25
98/98 [==============================] - 421s 4s/step - loss: 0.0995 - accuracy: 0.97
Epoch 16/25
98/98 [==============================] - 422s 4s/step - loss: 0.1147 - accuracy: 0.96
Epoch 17/25
```

```
98/98 [==============================] - 423s 4s/step - loss: 0.1107 - accuracy: 0.97
Epoch 18/25
98/98 [==============================] - 423s 4s/step - loss: 0.0688 - accuracy: 0.98
Epoch 19/25
98/98 [==============================] - 424s 4s/step - loss: 0.0563 - accuracy: 0.98
Epoch 20/25
98/98 [==============================] - 426s 4s/step - loss: 0.0415 - accuracy: 0.99
Epoch 21/25
98/98 [==============================] - 425s 4s/step - loss: 0.0386 - accuracy: 0.99
Epoch 22/25
98/98 [==============================] - 425s 4s/step - loss: 0.0600 - accuracy: 0.98
Epoch 23/25
98/98 [==============================] - 425s 4s/step - loss: 0.0519 - accuracy: 0.98
Epoch 24/25
98/98 [==============================] - 432s 4s/step - loss: 0.1167 - accuracy: 0.96
Epoch 25/25
98/98 [==============================] - 426s 4s/step - loss: 0.0424 - accuracy: 0.98
```

8. Save The Model

```
from tensorflow.keras.models import load_model

model.save('/content/drive/MyDrive/IBM/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/IBM/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/Car_damage/level/training/01-minor/0008.JPEG"
image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 1s 536ms/step
side
```

FOR LEVEL DAMAGE

IMAGE PRE PROCESSING

1. Import The ImageDataGenerator Library

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

2. 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)
```

3. Apply ImageDataGenerator Functionality To Trainset And Testset

```
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/Car_damage/level/
                                                 target_size = (224, 224),
                                                 batch_size = 10,
                                                 class_mode = 'categorical')
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/Car_damage/level/train
                                            target_size = (224, 224),
                                            batch_size = 10,
                                            class_mode = 'categorical')
```

```
Found 979 images belonging to 3 classes.
Found 979 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/Car_damage/level/training'
valid_path = '/content/drive/MyDrive/Car_damage/level/training'


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

3. Adding Flatten Layer

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


folders = glob('/content/drive/MyDrive/Car_damage/level/training/*')


folders
```

```
['/content/drive/MyDrive/Car_damage/body/training/02-side',
 '/content/drive/MyDrive/Car_damage/body/training/01-rear',
 '/content/drive/MyDrive/Car_damage/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.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
_________________________________________________________________
```

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.f

Epoch 1/25
98/98 [==============================] - 615s 6s/step - loss: 1.2465 - accuracy: 0.55
Epoch 2/25
98/98 [==============================] - 604s 6s/step - loss: 0.6654 - accuracy: 0.75
Epoch 3/25
98/98 [==============================] - 604s 6s/step - loss: 0.5950 - accuracy: 0.76
Epoch 4/25
98/98 [==============================] - 601s 6s/step - loss: 0.4964 - accuracy: 0.80
Epoch 5/25
98/98 [==============================] - 603s 6s/step - loss: 0.3559 - accuracy: 0.86
Epoch 6/25
98/98 [==============================] - 604s 6s/step - loss: 0.2425 - accuracy: 0.91
Epoch 7/25
98/98 [==============================] - 604s 6s/step - loss: 0.1964 - accuracy: 0.93
Epoch 8/25
98/98 [==============================] - 598s 6s/step - loss: 0.2119 - accuracy: 0.92
Epoch 9/25
98/98 [==============================] - 597s 6s/step - loss: 0.1111 - accuracy: 0.96
Epoch 10/25
98/98 [==============================] - 595s 6s/step - loss: 0.1394 - accuracy: 0.94
Epoch 11/25
98/98 [==============================] - 598s 6s/step - loss: 0.1167 - accuracy: 0.96
Epoch 12/25
98/98 [==============================] - 598s 6s/step - loss: 0.0823 - accuracy: 0.97
Epoch 13/25
98/98 [==============================] - 602s 6s/step - loss: 0.1062 - accuracy: 0.96
Epoch 14/25
98/98 [==============================] - 599s 6s/step - loss: 0.0717 - accuracy: 0.97
Epoch 15/25
98/98 [==============================] - 598s 6s/step - loss: 0.0692 - accuracy: 0.98
Epoch 16/25
98/98 [==============================] - 595s 6s/step - loss: 0.0449 - accuracy: 0.98
Epoch 17/25
98/98 [==============================] - 609s 6s/step - loss: 0.0522 - accuracy: 0.98
Epoch 18/25
98/98 [==============================] - 607s 6s/step - loss: 0.0386 - accuracy: 0.99
Epoch 19/25
98/98 [==============================] - 595s 6s/step - loss: 0.0381 - accuracy: 0.99
Epoch 20/25
98/98 [==============================] - 596s 6s/step - loss: 0.0196 - accuracy: 1.00
```

```
Epoch 21/25
98/98 [==============================] - 597s 6s/step - loss: 0.0394 - accuracy: 0.9
Epoch 22/25
98/98 [==============================] - 595s 6s/step - loss: 0.0377 - accuracy: 0.9
Epoch 23/25
98/98 [==============================] - 595s 6s/step - loss: 0.0387 - accuracy: 0.9
Epoch 24/25
98/98 [==============================] - 596s 6s/step - loss: 0.0279 - accuracy: 0.9
Epoch 25/25
98/98 [==============================] - 603s 6s/step - loss: 0.0271 - accuracy: 0.9
```

8. Save The Model

```
from tensorflow.keras.models import load_model

model.save('/content/drive/MyDrive/IBM/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/IBM/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/Car_damage/level/validation/01-minor/0002.JPEG"
image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 1s 510ms/step
severe
```

Colab paid products - Cancel contracts here

✓ 1s completed at 4:45 PM