```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": [],
      "collapsed_sections": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    },
    "accelerator": "GPU",
    "gpuClass": "standard"
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "\n",
        "## 1. Import The ImageDataGenerator Library"
      ],
      "metadata": {
        "id": "OLwXYl0cZNDJ"
      }
    },
    {
      "cell_type": "code",
      "execution_count": 2,
      "metadata": {
        "id": "YSAb5um_NjyD"
      },
      "outputs": [],
      "source": [
        "from tensorflow.keras.preprocessing.image import
ImageDataGenerator"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "## 2. Configure ImageDataGenerator Class\n",
        "## Image Data Augmentation"
      ],
      "metadata": {
        "id": "-hpvO-K2ZdOg"
      }
    },
    {
      "cell_type": "code",
      "source": [
```

```
        "train_datagen = ImageDataGenerator(rescale = 1./255,\n",
        "shear_range = 0.1,\n",
        "zoom_range = 0.1,\n",
        "horizontal_flip = True)\n",
        "test_datagen = ImageDataGenerator(rescale = 1./255)"
      ],
      "metadata": {
        "id": "DHLRuF6XQEk0"
      },
      "execution_count": 3,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "## 3. Apply ImageDataGenerator Functionality To Trainset And
Testset"
      ],
      "metadata": {
        "id": "hPuk6i36Z9gN"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/body/training',
target_size = (224, 224),batch_size = 10,class_mode = 'categorical')\n",
        "test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/body/validation'
,target_size = (224, 224),batch_size = 10,class_mode = 'categorical')"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "fCbU8vVHRecA",
        "outputId": "33fee3df-1b26-4efe-eabf-eb3aa1c0cbfe"
      },
      "execution_count": 4,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Found 979 images belonging to 3 classes.\n",
            "Found 171 images belonging to 3 classes.\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
```

```
        "## MODEL BUILDING\n",
        "### 1. Importing The Model Building Libraries"
      ],
      "metadata": {
        "id": "sUevHNoKaEFv"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import tensorflow as tf\n",
        "from tensorflow.keras.layers import Input, Lambda, Dense,
Flatten\n",
        "from tensorflow.keras.models import Model\n",
        "from tensorflow.keras.applications.vgg16 import VGG16\n",
        "from tensorflow.keras.applications.vgg19 import VGG19\n",
        "from tensorflow.keras.preprocessing import image\n",
        "from tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img\n",
        "from tensorflow.keras.models import Sequential\n",
        "import numpy as np\n",
        "from glob import glob"
      ],
      "metadata": {
        "id": "IOVnz2KKTm0U"
      },
      "execution_count": 5,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 2. Loading The Model"
      ],
      "metadata": {
        "id": "hrUsyRz9aMOI"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "IMAGE_SIZE = [224, 224]\n",
        "train_path = '/content/drive/MyDrive/body/training'\n",
        "valid_path = '/content/drive/MyDrive/body/validation'\n",
        "vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)"
      ],
      "metadata": {
        "id": "Q9f_p-MrTpWT",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "fc4ab32c-4135-4c58-ef61-a45dbe0c79ef"
      },
```

```
      "execution_count": 6,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Downloading data from
https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5\n",
            "58889256/58889256 [==============================] - 0s
0us/step\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 3. Adding Flatten Layer"
      ],
      "metadata": {
        "id": "iFgb_8K4aSgY"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "for layer in vgg16.layers:layer.trainable = False"
      ],
      "metadata": {
        "id": "P6jE5lGtUYql"
      },
      "execution_count": 7,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "folders = glob('/content/drive/MyDrive/body/training/*')\n",
        "folders"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "lHSZpSGyUcnW",
        "outputId": "c6339643-7cc7-4321-809a-bdb3b13f609e"
      },
      "execution_count": 8,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
```

```
          "['/content/drive/MyDrive/body/training/02-side',\n",
          " '/content/drive/MyDrive/body/training/00-front',\n",
          " '/content/drive/MyDrive/body/training/01-rear']"
        ]
      },
      "metadata": {},
      "execution_count": 8
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "x = Flatten()(vgg16.output)\n",
    "len(folders)"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "zyDGemnUUcxU",
    "outputId": "a36c2ff4-7635-41f5-bfcb-59c9305b41a8"
  },
  "execution_count": 9,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "3"
        ]
      },
      "metadata": {},
      "execution_count": 9
    }
  ]
},
{
  "cell_type": "markdown",
  "source": [
    "### 4. Adding Output Layer"
  ],
  "metadata": {
    "id": "KKtFLymtaYqF"
  }
},
{
  "cell_type": "code",
  "source": [
    "prediction = Dense(len(folders), activation='softmax')(x)"
  ],
  "metadata": {
    "id": "OicsUEzhVZTU"
  },
```

```
      "execution_count": 10,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 5. Creating A Model Object"
      ],
      "metadata": {
        "id": "HQHijmn6acuD"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model = Model(inputs=vgg16.input, outputs=prediction)"
      ],
      "metadata": {
        "id": "uOd8hi0jVl8L"
      },
      "execution_count": 11,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model.summary()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "FYkuPEUAVp-c",
        "outputId": "b77ab3cd-a02b-4a77-8581-bba24332ac64"
      },
      "execution_count": 12,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Model: \"model\"\n",
"_________________________________________________________________\n",
            " Layer (type)                Output Shape              Param
#   \n",
"=================================================================\n",
            " input_1 (InputLayer)        [(None, 224, 224, 3)]     0
\n",
            "
\n",
            " block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
\n",
```

```
            "
\n",
            " block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
\n",
            "
\n",
            " block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
\n",
            "
\n",
            " block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
\n",
            "
\n",
            " block2_conv2 (Conv2D)       (None, 112, 112, 128)
147584    \n",
            "
\n",
            " block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
\n",
            "
\n",
            " block3_conv1 (Conv2D)       (None, 56, 56, 256)
295168    \n",
            "
\n",
            " block3_conv2 (Conv2D)       (None, 56, 56, 256)
590080    \n",
            "
\n",
            " block3_conv3 (Conv2D)       (None, 56, 56, 256)
590080    \n",
            "
\n",
            " block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
\n",
            "
\n",
            " block4_conv1 (Conv2D)       (None, 28, 28, 512)
1180160   \n",
            "
\n",
            " block4_conv2 (Conv2D)       (None, 28, 28, 512)
2359808   \n",
            "
\n",
            " block4_conv3 (Conv2D)       (None, 28, 28, 512)
2359808   \n",
            "
\n",
            " block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
\n",
            "
\n",
```

```
            " block5_conv1 (Conv2D)       (None, 14, 14, 512)
2359808   \n",
            "
\n",
            " block5_conv2 (Conv2D)       (None, 14, 14, 512)
2359808   \n",
            "
\n",
            " block5_conv3 (Conv2D)       (None, 14, 14, 512)
2359808   \n",
            "
\n",
            " block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
\n",
            "
\n",
            " flatten (Flatten)           (None, 25088)             0
\n",
            "
\n",
            " dense (Dense)               (None, 3)                 75267
\n",
            "
\n",

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

"_________________________________________________________________\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 6. Configure The Learning Process"
      ],
      "metadata": {
        "id": "Opsej8T1ak-R"
      }
    },
    {
      "cell_type": "code",
      "source": [

"model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=[
'accuracy'])"
      ],
      "metadata": {
        "id": "rOoSqYKCVu3U"
      },
```

```
      "execution_count": 13,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 7. Train The Model"
      ],
      "metadata": {
        "id": "94SefBLBarUQ"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "r = model.fit_generator(\n",
        "training_set,\n",
        "validation_data=test_set,\n",
        "epochs=5,\n",
        "steps_per_epoch=len(training_set),\n",
        "validation_steps=len(test_set)\n",
        ")"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "jqFDRyYMao5u",
        "outputId": "740cbafa-f14e-4956-fd46-c10f5b88106e"
      },
      "execution_count": 21,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Epoch 1/5\n"
          ]
        },
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "/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.\n",
            "  \n"
          ]
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
```

```
            "98/98 [==============================] - 292s 3s/step -
loss: 0.9919 - accuracy: 0.5863 - val_loss: 0.9702 - val_accuracy:
0.6550\n",
            "Epoch 2/5\n",
            "98/98 [==============================] - 14s 144ms/step -
loss: 0.6779 - accuracy: 0.7354 - val_loss: 1.0075 - val_accuracy:
0.6374\n",
            "Epoch 3/5\n",
            "98/98 [==============================] - 14s 145ms/step -
loss: 0.5014 - accuracy: 0.8029 - val_loss: 1.8027 - val_accuracy:
0.4971\n",
            "Epoch 4/5\n",
            "98/98 [==============================] - 15s 148ms/step -
loss: 0.3898 - accuracy: 0.8580 - val_loss: 0.9130 - val_accuracy:
0.6959\n",
            "Epoch 5/5\n",
            "98/98 [==============================] - 15s 154ms/step -
loss: 0.2440 - accuracy: 0.9224 - val_loss: 1.0812 - val_accuracy:
0.6901\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 8. Save The Model"
      ],
      "metadata": {
        "id": "0J14BGt9bBVJ"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.models import load_model\n",
        "model.save('/content/drive/MyDrive/ibm project/Intelligent
Vehicle Damage Assessment & Cost Estimator/MODEL/BODY.h5')"
      ],
      "metadata": {
        "id": "PBJFfRl9bF_l"
      },
      "execution_count": 15,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 9. Test The Model"
      ],
      "metadata": {
        "id": "Nl8gxE73bPe7"
      }
    },
```

```
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.models import load_model\n",
        "import cv2\n",
        "from skimage.transform import resize"
      ],
      "metadata": {
        "id": "16Cyy5cQbPtD"
      },
      "execution_count": 16,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model = load_model('/content/drive/MyDrive/ibm
project/Intelligent Vehicle Damage Assessment & Cost
Estimator/MODEL/BODY.h5')"
      ],
      "metadata": {
        "id": "moxadPHxbjI1"
      },
      "execution_count": 17,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "def detect(frame):\n",
        "  img = cv2.resize(frame,(224,224))\n",
        "  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)\n",
        "  if(np.max(img)>1):\n",
        "    img = img/255.0\n",
        "    img = np.array([img])\n",
        "    prediction = model.predict(img)\n",
        "    label = [\"front\",\"rear\",\"side\"]\n",
        "    preds = label[np.argmax(prediction)]\n",
        "    return preds"
      ],
      "metadata": {
        "id": "thcCd1e6bz29"
      },
      "execution_count": 18,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "data = \"/content/drive/MyDrive/body/training/00-
front/0007.JPEG\"\n",
        "image = cv2.imread(data)\n",
        "print(detect(image))"
      ],
```

```
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "3lUBWrJjb8D_",
        "outputId": "bb08e900-0568-4377-95bd-068e407528a7"
      },
      "execution_count": 19,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "1/1 [==============================] - 1s 812ms/step\n",
            "front\n"
          ]
        }
      ]
    }
  ]
}
```