```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    },
    "gpuClass": "standard",
    "accelerator": "GPU"
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "## IMAGE PRE PROCESSING\n",
        "### 1. Import The ImageDataGenerator Library"
      ],
      "metadata": {
        "id": "B7sKH_J5dCMW"
      }
    },
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "id": "9Kg3-zJbc93E"
      },
      "outputs": [],
      "source": [
        "from tensorflow.keras.preprocessing.image import 
ImageDataGenerator"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 1. Configure ImageDataGenerator Class\n"
      ],
      "metadata": {
        "id": "KdFjEvMrdUQk"
      }
    },
    {
      "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": "BbsFcw4Mdd5m"
      },
      "execution_count": 2,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 2. Apply ImageDataGenerator Functionality To Trainset
And Testset"
      ],
      "metadata": {
        "id": "bDOrsoh4dv4Q"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/level/train
ing',target_size = (224, 224),batch_size = 10,class_mode =
'categorical')\n",
        "test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/level/valida
tion',target_size = (224, 224),batch_size = 10,class_mode =
'categorical')"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "kafmaOH-d5JR",
        "outputId": "9f8bbd9e-36f0-4eb2-f8eb-79c2e6e7251a"
      },
      "execution_count": 3,
      "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": "HGQD2XJJezvM"
      }
    },
    {
      "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": "2pXinG6Ve6I6"
      },
      "execution_count": 4,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "## 2. Loading The Model"
      ],
      "metadata": {
        "id": "Es3CYt5Ci4KM"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "IMAGE_SIZE = [224, 224]\n",
        "train_path = '/content/drive/MyDrive/level/training'\n",
        "valid_path = '/content/drive/MyDrive/level/validation'"
      ],
      "metadata": {
        "id": "JIg9GA8ri9oa"
      },
      "execution_count": 5,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 3. Adding Flatten Layer"
      ],
      "metadata": {
        "id": "fCGeJlcSj5UN"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "vgg16 = VGG16(input_shape=IMAGE_SIZE + [3],
weights='imagenet', include_top=False)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "5o5W8rWlkafP",
        "outputId": "a743edd7-5a53-4ff9-cfc5-7b3b37bfa6b5"
      },
      "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 [==============================] - 2s
0us/step\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "for layer in vgg16.layers:layer.trainable = False\n",
        "folders =
glob('/content/drive/MyDrive/level/training/*')\n",
        "folders"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "QB0zQlQFj9EI",
        "outputId": "d710ae7f-c009-4d53-fd5c-a628eb012be8"
      },
      "execution_count": 7,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "['/content/drive/MyDrive/level/training/02-
moderate',\n",
              " '/content/drive/MyDrive/level/training/03-
severe',\n",
              " '/content/drive/MyDrive/level/training/01-minor']"
            ]
          },
          "metadata": {},
          "execution_count": 7
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "x = Flatten()(vgg16.output)\n",
        "len(folders)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "xSlQzZ73kzSL",
        "outputId": "9514f47b-b654-486e-bb60-16f0d9104ed0"
      },
      "execution_count": 8,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "3"
            ]
          },
          "metadata": {},
          "execution_count": 8
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 4. Adding Output Layer"
      ],
      "metadata": {
        "id": "kkwiwoQZk7Kl"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "prediction = Dense(len(folders), activation='softmax')(x)"
      ],
      "metadata": {
        "id": "hNCnik_Dk9Px"
      },
      "execution_count": 9,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 5. Creating A Model Object"
      ],
      "metadata": {
        "id": "L0-5H7yUlBZT"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model = Model(inputs=vgg16.input, outputs=prediction)"
      ],
      "metadata": {
        "id": "pev9HmY3lBkw"
      },
      "execution_count": 10,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model.summary()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "gxORtqV4lN_C",
        "outputId": "1a4b9b9b-cc29-4f57-ef63-38725489c865"
      },
      "execution_count": 11,
      "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": "DGscyME1lLyt"
  }
},
{
  "cell_type": "code",
  "source": [
    "model.compile(\n",
    "loss='categorical_crossentropy',\n",
    "optimizer='adam',\n",
    "metrics=['accuracy']\n",
    ")"
  ],
  "metadata": {
    "id": "BULeM8ZYlVjB"
  },
  "execution_count": 12,
  "outputs": []
},
{
  "cell_type": "markdown",
  "source": [
    "### 7. Train The Model"
  ],
  "metadata": {
    "id": "g_oS81iVlZwI"
  }
},
{
  "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": "oKTjkrqPlZ9e",
        "outputId": "123bf3a9-e4a8-49a7-95f7-5004896d1714"
      },
      "execution_count": 13,
      "outputs": [
        {
          "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": [
            "Epoch 1/5\n",
            "98/98 [==============================] - 407s 4s/step -
loss: 1.2409 - accuracy: 0.5628 - val_loss: 1.2019 - val_accuracy:
0.5614\n",
            "Epoch 2/5\n",
            "98/98 [==============================] - 18s 179ms/step
- loss: 0.7316 - accuracy: 0.7191 - val_loss: 0.9586 - val_accuracy:
0.6082\n",
            "Epoch 3/5\n",
            "98/98 [==============================] - 16s 164ms/step
- loss: 0.5469 - accuracy: 0.7957 - val_loss: 1.0207 - val_accuracy:
0.6140\n",
            "Epoch 4/5\n",
            "98/98 [==============================] - 16s 167ms/step
- loss: 0.4278 - accuracy: 0.8223 - val_loss: 1.6515 - val_accuracy:
0.5965\n",
            "Epoch 5/5\n",
            "98/98 [==============================] - 17s 177ms/step
- loss: 0.4449 - accuracy: 0.8284 - val_loss: 1.2299 - val_accuracy:
0.6199\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 8. Save The Model"
      ],
      "metadata": {
        "id": "X0VPatXFl1jg"
      }
    },
    {
      "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/LEVEL.h5')"
      ],
      "metadata": {
        "id": "EPT6bkyyl3WD"
      },
      "execution_count": 14,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "### 9. Test The Model"
      ],
      "metadata": {
        "id": "XH69XsO3mIum"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.models import load_model\n",
        "import cv2\n",
        "from skimage.transform import resize"
      ],
      "metadata": {
        "id": "LmolQnm5mLqm"
      },
      "execution_count": 15,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model = load_model('/content/drive/MyDrive/ibm
project/Intelligent Vehicle Damage Assessment & Cost
Estimator/MODEL/LEVEL.h5')"
      ],
      "metadata": {
        "id": "LhXWyCfjmUSA"
      },
      "execution_count": 16,
      "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 = [\"minor\",\"moderate\",\"severe\"]\n",
        "    preds = label[np.argmax(prediction)]\n",
        "    return preds"
      ],
      "metadata": {
        "id": "RjsDOmLImUcd"
      },
      "execution_count": 17,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "data = \"/content/drive/MyDrive/level/training/01-
minor/0007.JPEG\"\n",
        "image = cv2.imread(data)\n",
        "print(detect(image))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "lMGtNXRImo3q",
        "outputId": "4a7e0d75-b229-47fa-cde2-a0a79f87f9a4"
      },
      "execution_count": 18,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "1/1 [==============================] - 0s 157ms/step\n",
            "minor\n"
          ]
        }
      ]
    }
  ]
}
```