```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": [],
      "collapsed_sections": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "# **Assignment 3**"
      ],
      "metadata": {
        "id": "j8bF22iNxMDx"
      }
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **CNN MODEL FOR FLOWER CLASSIFICATION**\n",
        "# **Trained by Team ID : PNT2022TMID17050**"
      ],
      "metadata": {
        "id": "BdPpvl8Q0IlY"
      }
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Pre-Requisites**"
      ],
      "metadata": {
        "id": "d5mO5lxMq9BQ"
      }
    },
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
```

```
          "base_uri": "https://localhost:8080/"
        },
        "id": "h4I4dfJ6p3kx",
        "outputId": "5605b9f7-ebc0-4587-88c6-268194f1335d"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Drive already mounted at /content/drive; to attempt to
forcibly remount, call drive.mount(\"/content/drive\",
force_remount=True).\n"
          ]
        }
      ],
      "source": [
        "from google.colab import drive\n",
        "drive.mount('/content/drive')"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **STEP 1 UNZIP FILES**"
      ],
      "metadata": {
        "id": "B4Kss5zOrXUz"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "cd/content/drive/MyDrive/AI_IBM"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "DRNhAxfvrWKc",
        "outputId": "f1a087a8-6f33-4d81-ff79-e50f4c1ff623"
      },
      "execution_count": 2,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "/content/drive/MyDrive/AI_IBM\n"
          ]
```

```
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "!unzip Flowers-Dataset.zip"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "B_hJ27NKrhKz",
        "outputId": "9b319781-61d1-4a0e-ea93-f279d067bfc7"
      },
      "execution_count": 3,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Archive:  Flowers-Dataset.zip\n",
            "replace flowers/daisy/100080576_f52e8ee070_n.jpg? [y]es,
[n]o, [A]ll, [N]one, [r]ename: N\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **STEP 2 Image** **Augumentation**"
      ],
      "metadata": {
        "id": "hvG8h70rrphq"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.preprocessing.image import
ImageDataGenerator"
      ],
      "metadata": {
        "id": "itQt2Ad8rtk8"
      },
      "execution_count": 4,
      "outputs": []
    },
    {
```

```
      "cell_type": "code",
      "source": [
        "train_datagen=ImageDataGenerator(rescale=1./255,
zoom_range=0.2,horizontal_flip=True,vertical_flip=False)"
      ],
      "metadata": {
        "id": "9yZUiTxnr0UN"
      },
      "execution_count": 5,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "test_datagen=ImageDataGenerator(rescale=1./255)"
      ],
      "metadata": {
        "id": "zD7ristVr3F3"
      },
      "execution_count": 6,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [

"x_train=train_datagen.flow_from_directory(r\"/content/drive/MyDrive/
AI_IBM/
flowers\",target_size=(64,64),class_mode='categorical',batch_size=24)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "BjQo5zGHuHN4",
        "outputId": "d3d1e296-e74d-4e52-cce8-8d26459d10f1"
      },
      "execution_count": 7,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Found 4317 images belonging to 5 classes.\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
```

```
      "source": [
"x_test=test_datagen.flow_from_directory(r\"/content/drive/MyDrive/
AI_IBM/
flowers\",target_size=(64,64),class_mode='categorical',batch_size=24)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "e4YJwWrCukDq",
        "outputId": "e71a3e44-6642-4592-fa96-7af9c6edb08f"
      },
      "execution_count": 8,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Found 4317 images belonging to 5 classes.\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "x_train.class_indices"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "EgBhHHYTuv4X",
        "outputId": "8a9f62e0-7d2b-4138-c5ce-4ca16b78fbd1"
      },
      "execution_count": 9,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3,
'tulip': 4}"
            ]
          },
          "metadata": {},
          "execution_count": 9
        }
      ]
```

```
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Step -3 Initializing CNN And Create Model**"
      ],
      "metadata": {
        "id": "O5cz-9qOJM_s"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.models import Sequential\n",
        "from tensorflow.keras.layers import
Dense,Convolution2D,MaxPooling2D,Flatten"
      ],
      "metadata": {
        "id": "QAUHi2otRcoC"
      },
      "execution_count": 10,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Step -4 Add layers**"
      ],
      "metadata": {
        "id": "xew7skua3a0z"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model=Sequential()"
      ],
      "metadata": {
        "id": "dack9NXYR2t6"
      },
      "execution_count": 11,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **4.1 Input Layers (Convolution ,MaxPooling,Flatten)**"
      ],
      "metadata": {
        "id": "SzIvL8Q52DFR"
```

```
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model.add(Convolution2D(32,
(3,3),input_shape=(64,64,3),activation='relu'))"
      ],
      "metadata": {
        "id": "qPUbKxHGR7EX"
      },
      "execution_count": 12,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model.add(MaxPooling2D(pool_size=(2,2)))"
      ],
      "metadata": {
        "id": "IBGMZ7sSSAIB"
      },
      "execution_count": 13,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model.add(Flatten())"
      ],
      "metadata": {
        "id": "c65fXm9KSErL"
      },
      "execution_count": 14,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "model.summary()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "-go5E-VbSIau",
        "outputId": "1f46f35d-1950-4456-bce9-16a06053d40f"
      },
      "execution_count": 15,
      "outputs": [
```

```
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Model: \"sequential\"\n",
"_________________________________________________________________\n",
            " Layer (type)                Output Shape              
Param #   \n",
"=================================================================\n",
            " conv2d (Conv2D)             (None, 62, 62, 32)        
896       \n",
            "
\n",
            " max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
\n",
            " )
\n",
            "
\n",
            " flatten (Flatten)           (None, 30752)             0
\n",
            "
\n",
"=================================================================\n",
            "Total params: 896\n",
            "Trainable params: 896\n",
            "Non-trainable params: 0\n",
"_________________________________________________________________\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **4.2 Hidden Layers**"
      ],
      "metadata": {
        "id": "f4ZSQPIFJaeb"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model.add(Dense(300,activation='relu'))\n",
        "model.add(Dense(150,activation='relu'))"
```

```
      ],
      "metadata": {
        "id": "x8MIUG1PSZ21"
      },
      "execution_count": 16,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **4.3 Output Layers**"
      ],
      "metadata": {
        "id": "PNLk8KHHJf3K"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model.add(Dense(5,activation='softmax'))"
      ],
      "metadata": {
        "id": "grI0IbuwSeqO"
      },
      "execution_count": 17,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [

"model.compile(loss='categorical_crossentropy',optimizer='adam',metric
s=['accuracy'])"
      ],
      "metadata": {
        "id": "l44vMW4QShaw"
      },
      "execution_count": 18,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "len(x_train)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "Beiar4NESkL4",
```

```
        "outputId": "4b264b09-51b5-4786-b2a8-d60ac129229d"
      },
      "execution_count": 19,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "180"
            ]
          },
          "metadata": {},
          "execution_count": 19
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Step -5 Train the Model**"
      ],
      "metadata": {
        "id": "Y9f3ElSv3Nc6"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model.fit_generator(x_train,steps_per_epoch=len(x_train),
validation_data=x_test, validation_steps=len(x_test), epochs= 30)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "ATt0m5Cv6R-w",
        "outputId": "734d2b05-c864-450f-a46f-8ce129904306"
      },
      "execution_count": 20,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [

"/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.\n",
            "  \"\"\"Entry point for launching an IPython kernel.\n"
```

```
        ]
      },
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "Epoch 1/30\n",
          "180/180 [==============================] - 393s 2s/step -
loss: 1.3213 - accuracy: 0.4714 - val_loss: 1.1275 - val_accuracy:
0.5532\n",
          "Epoch 2/30\n",
          "180/180 [==============================] - 74s 409ms/step
- loss: 1.0600 - accuracy: 0.5854 - val_loss: 0.9406 - val_accuracy:
0.6301\n",
          "Epoch 3/30\n",
          "180/180 [==============================] - 73s 405ms/step
- loss: 0.9678 - accuracy: 0.6247 - val_loss: 0.9603 - val_accuracy:
0.6203\n",
          "Epoch 4/30\n",
          "180/180 [==============================] - 77s 429ms/step
- loss: 0.8884 - accuracy: 0.6546 - val_loss: 0.8187 - val_accuracy:
0.6938\n",
          "Epoch 5/30\n",
          "180/180 [==============================] - 76s 422ms/step
- loss: 0.8358 - accuracy: 0.6787 - val_loss: 0.7393 - val_accuracy:
0.7225\n",
          "Epoch 6/30\n",
          "180/180 [==============================] - 75s 418ms/step
- loss: 0.7924 - accuracy: 0.6965 - val_loss: 0.8389 - val_accuracy:
0.6928\n",
          "Epoch 7/30\n",
          "180/180 [==============================] - 73s 405ms/step
- loss: 0.7521 - accuracy: 0.7158 - val_loss: 0.8503 - val_accuracy:
0.6789\n",
          "Epoch 8/30\n",
          "180/180 [==============================] - 74s 411ms/step
- loss: 0.7048 - accuracy: 0.7313 - val_loss: 0.6492 - val_accuracy:
0.7521\n",
          "Epoch 9/30\n",
          "180/180 [==============================] - 72s 400ms/step
- loss: 0.6502 - accuracy: 0.7521 - val_loss: 0.6458 - val_accuracy:
0.7438\n",
          "Epoch 10/30\n",
          "180/180 [==============================] - 74s 409ms/step
- loss: 0.6182 - accuracy: 0.7684 - val_loss: 0.5721 - val_accuracy:
0.7818\n",
          "Epoch 11/30\n",
          "180/180 [==============================] - 72s 402ms/step
- loss: 0.5662 - accuracy: 0.7931 - val_loss: 0.5968 - val_accuracy:
0.7725\n",
```

```
            "Epoch 12/30\n",
            "180/180 [==============================] - 72s 401ms/step
- loss: 0.5600 - accuracy: 0.7908 - val_loss: 0.6907 - val_accuracy:
0.7612\n",
            "Epoch 13/30\n",
            "180/180 [==============================] - 72s 399ms/step
- loss: 0.5064 - accuracy: 0.8138 - val_loss: 0.5185 - val_accuracy:
0.8117\n",
            "Epoch 14/30\n",
            "180/180 [==============================] - 71s 394ms/step
- loss: 0.4830 - accuracy: 0.8249 - val_loss: 0.3613 - val_accuracy:
0.8673\n",
            "Epoch 15/30\n",
            "180/180 [==============================] - 71s 397ms/step
- loss: 0.4650 - accuracy: 0.8196 - val_loss: 0.3396 - val_accuracy:
0.8768\n",
            "Epoch 16/30\n",
            "180/180 [==============================] - 71s 393ms/step
- loss: 0.4117 - accuracy: 0.8559 - val_loss: 0.3472 - val_accuracy:
0.8738\n",
            "Epoch 17/30\n",
            "180/180 [==============================] - 71s 397ms/step
- loss: 0.3892 - accuracy: 0.8631 - val_loss: 0.3314 - val_accuracy:
0.8826\n",
            "Epoch 18/30\n",
            "180/180 [==============================] - 70s 389ms/step
- loss: 0.3441 - accuracy: 0.8726 - val_loss: 0.4008 - val_accuracy:
0.8589\n",
            "Epoch 19/30\n",
            "180/180 [==============================] - 73s 404ms/step
- loss: 0.3467 - accuracy: 0.8719 - val_loss: 0.2484 - val_accuracy:
0.9060\n",
            "Epoch 20/30\n",
            "180/180 [==============================] - 72s 398ms/step
- loss: 0.3327 - accuracy: 0.8758 - val_loss: 0.2234 - val_accuracy:
0.9210\n",
            "Epoch 21/30\n",
            "180/180 [==============================] - 73s 403ms/step
- loss: 0.2807 - accuracy: 0.9009 - val_loss: 0.2830 - val_accuracy:
0.9036\n",
            "Epoch 22/30\n",
            "180/180 [==============================] - 70s 392ms/step
- loss: 0.2751 - accuracy: 0.9013 - val_loss: 0.2392 - val_accuracy:
0.9141\n",
            "Epoch 23/30\n",
            "180/180 [==============================] - 73s 404ms/step
- loss: 0.2549 - accuracy: 0.9097 - val_loss: 0.2221 - val_accuracy:
0.9189\n",
            "Epoch 24/30\n",
            "180/180 [==============================] - 72s 399ms/step
```

```
- loss: 0.2412 - accuracy: 0.9243 - val_loss: 0.2029 - val_accuracy:
0.9291\n",
            "Epoch 25/30\n",
            "180/180 [==============================] - 72s 402ms/step
- loss: 0.2360 - accuracy: 0.9199 - val_loss: 0.1965 - val_accuracy:
0.9307\n",
            "Epoch 26/30\n",
            "180/180 [==============================] - 72s 401ms/step
- loss: 0.2199 - accuracy: 0.9201 - val_loss: 0.1919 - val_accuracy:
0.9331\n",
            "Epoch 27/30\n",
            "180/180 [==============================] - 72s 400ms/step
- loss: 0.2008 - accuracy: 0.9363 - val_loss: 0.1218 - val_accuracy:
0.9560\n",
            "Epoch 28/30\n",
            "180/180 [==============================] - 73s 406ms/step
- loss: 0.1889 - accuracy: 0.9310 - val_loss: 0.2838 - val_accuracy:
0.9108\n",
            "Epoch 29/30\n",
            "180/180 [==============================] - 70s 389ms/step
- loss: 0.2046 - accuracy: 0.9275 - val_loss: 0.2116 - val_accuracy:
0.9307\n",
            "Epoch 30/30\n",
            "180/180 [==============================] - 70s 392ms/step
- loss: 0.1886 - accuracy: 0.9372 - val_loss: 0.2091 - val_accuracy:
0.9280\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "<keras.callbacks.History at 0x7f3e15438e50>"
            ]
          },
          "metadata": {},
          "execution_count": 20
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Step -6 Save The model**"
      ],
      "metadata": {
        "id": "1uK880jw9Kru"
      }
    },
    {
```

```
      "cell_type": "code",
      "source": [
        "model.save('Flowers_classification_model1.h5')"
      ],
      "metadata": {
        "id": "scoaKurE9FZk"
      },
      "execution_count": 21,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **Step -7 Test The model**"
      ],
      "metadata": {
        "id": "YAH2UVpi9RMV"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "ls"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "Z-co6hBAEmzg",
        "outputId": "bf8a661d-3210-4695-dcb7-48e6f365dfce"
      },
      "execution_count": 22,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "\u001b[0m\u001b[01;34mflowers\u001b[0m/
Flowers_classification_model1.h5  Flowers-Dataset.zip  video.mp4\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "import numpy as np\n",
        "from tensorflow.keras.models import load_model\n",
        "from tensorflow.keras.preprocessing import image"
      ],
```

```
      "metadata": {
        "id": "mJvRRo7VvkeO"
      },
      "execution_count": 23,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "# Load the model\n",
        "model=load_model('Flowers_classification_model1.h5')"
      ],
      "metadata": {
        "id": "xo6F_4jw9KBZ"
      },
      "execution_count": 24,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "img=image.load_img(r\"/content/s3.jpg\",target_size=(64,64))\
n",
        "x=image.img_to_array(img)\n",
        "x=np.expand_dims(x,axis=0)\n",
        "y=np.argmax(model.predict(x),axis=1)\n",
        "# x_train.class_indices\n",
        "index=['daisy','dandelion','rose','sunflower','tulip']\n",
        "index[y[0]]"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 35
        },
        "id": "2rnrfMAf-AB9",
        "outputId": "c6357a8b-5163-4884-c82e-05651a65571c"
      },
      "execution_count": 38,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "'sunflower'"
            ],
            "application/vnd.google.colaboratory.intrinsic+json": {
              "type": "string"
            }
          },
```

```
          "metadata": {},
          "execution_count": 38
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# **We Achieved 93 percent of accuracy with this model** \n",
        "# **Trained by Team ID : PNT2022TMID17050**"
      ],
      "metadata": {
        "id": "2f85wU8fL0Si"
      }
    }
  ]
}
```