```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# IBM Project Name: Real-Time Communication System Powered by AI for
Specially Abled\n",
    "# TEAM ID: PNT2022TMID34274\n",
    "# TEAM Member:ISWARYA I "
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# IBM WATSON STUDIO DEPLOYMENT CODE "
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 1.]INSTALLING THE KERAS ,INSTALLING THE TENSORFLOW"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 97,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "rNxm_bwDDmMj",
    "outputId": "e55c8ab9-8150-4d3e-ce51-bd273575630e",
    "scrolled": false
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Requirement already satisfied: Keras==2.2.4 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (2.2.4)\r\n",
      "Requirement already satisfied: h5py in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from Keras==2.2.4) (3.2.1)\r\n",
      "Requirement already satisfied: keras-preprocessing>=1.0.5 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
Keras==2.2.4) (1.1.2)\r\n",
      "Requirement already satisfied: numpy>=1.9.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
Keras==2.2.4) (1.20.3)\r\n",
      "Requirement already satisfied: pyyaml in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from Keras==2.2.4) (5.4.1)\r\n",
      "Requirement already satisfied: keras-applications>=1.0.6 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
Keras==2.2.4) (1.0.8)\r\n",
      "Requirement already satisfied: scipy>=0.14 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
Keras==2.2.4) (1.7.3)\r\n",
      "Requirement already satisfied: six>=1.9.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
Keras==2.2.4) (1.15.0)\r\n"
     ]
    }
   ],
   "source": [
    "!pip install Keras==2.2.4\n",
    "!pip install tensorflow==2.7"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "\n",
    "# 2.]IMPORTING LIBRARIES TO BUILD MODEL."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "dutY_IwFDl9g"
   },
   "outputs": [],
   "source": [
    "#library to train the model\n",
    "import keras\n",
    "import tensorflow\n",
    "\n",
    "\n",
    "from tensorflow.keras.models import Sequential\n",
    "from tensorflow.keras.layers import
Dense,Convolution2D,MaxPooling2D, Flatten"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 3.]IMPORTING LIBRARIES FOR IMAGE AUGMENTATION."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 99,
   "metadata": {
    "id": "3CSbv31FJuT1"
   },
   "outputs": [],
   "source": [
    "#image augmentation\n",
    "from tensorflow.keras.preprocessing.image import
ImageDataGenerator\n",
"train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,shear_ran
ge=0.2,horizontal_flip=True,vertical_flip=False)\n",
    "test_datagen=ImageDataGenerator(rescale=1./255)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 4.]ADDING STREAMING_BODY_OBJECT FOR DATASET.ZIP"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 100,
   "metadata": {},
   "outputs": [],
   "source": [
    "\n",
    "import os, types\n",
    "import pandas as pd\n",
    "from botocore.client import Config\n",
    "import ibm_boto3\n",
    "\n",
    "def __iter__(self): return 0\n",
    "\n",
    "# @hidden_cell\n",
    "# The following code accesses a file in your IBM Cloud Object
Storage. It includes your credentials.\n",
    "# You might want to remove those credentials before you share the
notebook.\n",
    "cos_client = ibm_boto3.client(service_name='s3',\n",
    "    ibm_api_key_id='aqprHZFuH38ECUn869hHk4qyvS_iKJfrZAWUJJQ-
mQKx',\n",
    "    ibm_auth_endpoint=\"https://iam.cloud.ibm.com/oidc/token\",\n",
    "    config=Config(signature_version='oauth'),\n",
    "    endpoint_url='https://s3.private.us.cloud-object-
storage.appdomain.cloud')\n",
    "\n",
    "bucket = 'realtimecommunicationforspecially-donotdelete-pr-
rfqndcvwgch6fu'\n",
    "object_key = 'Dataset.zip'\n",
    "\n",
    "streaming_body_4 = cos_client.get_object(Bucket=bucket,
Key=object_key)['Body']\n",
    "\n",
    "# Your data file was loaded into a botocore.response.StreamingBody
object.\n",
    "# Please read the documentation of ibm_boto3 and pandas to learn
more about the possibilities to load the data.\n",
    "# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-
python/\n",
    "# pandas documentation: http://pandas.pydata.org/\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 101,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "haMajSs9DliR",
    "outputId": "7451541d-41e1-4ba8-9c28-cf00eea03b0b"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "\u001b[0m\u001b[01;34mDataset\u001b[0m/
\u001b[01;34mtest_set\u001b[0m/  \u001b[01;34mtraining_set\u001b[0m/\r\n"
     ]
    }
   ],
   "source": [
    "ls"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "bqmiYuHxEe8t"
   },
   "source": [
    "# 5.]UNZIPPING THE DATASET"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 102,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 130
    },
    "id": "R8H9cP5PEhbj",
    "outputId": "ce76c8c9-6b37-4849-ea47-97dad4d231a7"
   },
   "outputs": [],
   "source": [
    "from io import BytesIO\n",
    "import zipfile\n",
    "unzip=zipfile.ZipFile(BytesIO(streaming_body_4.read()),'r')\n",
    "file_paths=unzip.namelist()\n",
    "for path in file_paths:\n",
    "    unzip.extract(path)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 103,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'/home/wsuser/work/Dataset'"
      ]
     },
     "execution_count": 103,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "pwd"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 104,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "L1-M6zotIpLy",
    "outputId": "2ae9caf5-a518-4b49-cb5b-e65f96842168"
   },
   "outputs": [],
   "source": [
    "#checking that the dataset is there are not\n",
    "import os\n",
    "filenamer = os.listdir('/home/wsuser/work/Dataset/training_set')"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 6.]TRAINING AND TESTING IMAGES UNDER CLASSES"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 105,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "fWBTCyOVKp01",
    "outputId": "c01cd057-5eae-429d-c737-541bb598118b"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Found 15750 images belonging to 9 classes.\n"
     ]
    }
   ],
   "source": [
"x_train=train_datagen.flow_from_directory(\"/home/wsuser/work/Dataset/tr
aining_set\",target_size=(64,64),class_mode=\"categorical\",batch_size=25
)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 106,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "yrG3iHrCKpRP",
    "outputId": "148c6bdd-fa51-4729-91e1-9ca60d5c7b5a"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Found 2250 images belonging to 9 classes.\n"
     ]
    }
   ],
   "source": [

"x_test=test_datagen.flow_from_directory(\"/home/wsuser/work/Dataset/test
_set\",target_size=(64,64),\n",
    "class_mode='categorical' , batch_size=25)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 7.]TOTAL CLASSES UNDER TRAINING AND TESTING."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 107,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "vjoAMqLiL2BV",
    "outputId": "2469327f-bc34-4811-9d6f-8d16e3ee57ff"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7,
'I': 8}"
      ]
     },
     "execution_count": 107,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "x_train.class_indices"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 108,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7,
'I': 8}"
      ]
     },
     "execution_count": 108,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "x_test.class_indices"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 109,
   "metadata": {
    "id": "yke48DPEKGm3"
   },
   "outputs": [],
   "source": [

"train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizonta
l_flip=True,vertical_flip=False)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 110,
   "metadata": {
    "id": "XglvnjXpKGTF"
   },
   "outputs": [],
   "source": [
    "test_datagen=ImageDataGenerator(rescale=1./255)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "Ska8jAKhMcjQ"
   },
    "source": [
     "# 8.]MODEL BUILDING USING CNN"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 111,
    "metadata": {
     "id": "b9q-J6A0ME4K"
    },
    "outputs": [],
    "source": [
     "model=Sequential()"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 112,
    "metadata": {
     "id": "jMemaPnZNUHz"
    },
    "outputs": [],
    "source": [

"model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation='relu'
))"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 113,
    "metadata": {
     "id": "FC-UXn7wNT6x"
    },
    "outputs": [],
    "source": [
     "model.add(MaxPooling2D(pool_size=(2,2)))"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 114,
    "metadata": {
     "id": "Bib-ZohnNTet"
    },
    "outputs": [],
    "source": [
     "model.add(Flatten())"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 115,
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "07-A3ymZNkOl",
    "outputId": "4158a17e-898d-4dd1-e3b0-2ae5927c2ae0"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Model: \"sequential_1\"\n",

"_________________________________________________________________\n",
      "Layer (type)                 Output Shape              Param #   
\n",

"=================================================================\n",
      "conv2d_1 (Conv2D)            (None, 62, 62, 32)        896       
\n",

"_________________________________________________________________\n",
      "max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0         
\n",

"_________________________________________________________________\n",
      "flatten_1 (Flatten)          (None, 30752)             0         
\n",

"=================================================================\n",
      "Total params: 896\n",
      "Trainable params: 896\n",
      "Non-trainable params: 0\n",

"_________________________________________________________________\n"
     ]
    }
   ],
   "source": [
    "model.summary()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "QZoyo7TtNj9u"
   },
   "source": [
    "# 9.]ADDING LAYERS FOR MODEL TRAINING."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "HLK-QpeFNwyz"
   },
   "source": [
    "# HIDDEN LAYERS"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 117,
   "metadata": {
    "id": "fYWVG08rNjwG"
   },
   "outputs": [],
   "source": [
    "model.add(Dense(units = 300, activation='relu'))\n",
    "#model.add(Dense(unit = 150,init = \"uniform\"
activation='softmax'))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "Qm4LWKnWN81_"
   },
   "source": [
    "# OUTPUT LAYERS"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 118,
   "metadata": {
    "id": "rCwPljf-NjgO"
   },
   "outputs": [],
   "source": [
    "model.add(Dense(units = 5, activation='softmax'))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 10.]OPTIMIZING THE MODEL "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 119,
   "metadata": {
    "id": "TlnJKIOGOD6t"
   },
   "outputs": [],
   "source": [

"model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=[
'accuracy'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 120,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "zLuzriYTODnO",
    "outputId": "117cf1c3-97af-4d83-bc0d-42f5dfa28682"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "630"
      ]
     },
     "execution_count": 120,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "\n",
    "len(x_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 121,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "90"
      ]
     },
     "execution_count": 121,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "len(x_test)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 11.]FITTING THE MODEL"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 125,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 1000
    },
    "id": "dDjZmKsWOPlc",
    "outputId": "989390bd-4c52-49c7-8408-ce22d8f4dfc3"
   },
   "outputs": [
    {
     "ename": "InvalidArgumentError",
     "evalue": " logits and labels must be broadcastable:
logits_size=[25,5] labels_size=[25,9]\n\t [[node
categorical_crossentropy/softmax_cross_entropy_with_logits\n (defined at
/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/backend.py:4889)\n]]
[Op:__inference_train_function_2383]\n\nErrors may have originated from
an input operation.\nInput Source operations connected to node
categorical_crossentropy/softmax_cross_entropy_with_logits:\nIn[0]
categorical_crossentropy/softmax_cross_entropy_with_logits/Reshape:\t\nIn
[1]
categorical_crossentropy/softmax_cross_entropy_with_logits/Reshape_1:\n\n
Operation defined at: (most recent call last)\n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/runpy.py\", line 197, in
_run_module_as_main\n>>>     return _run_code(code, main_globals,
None,\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/runpy.py\", line 87, in _run_code\n>>>     exec(code,
run_globals)\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/ipykernel/__main__.py\", line 3, in
<module>\n>>>     app.launch_new_instance()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/traitlets/config/application.py\", line 846, in
launch_instance\n>>>     app.start()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/kernelapp.py\", line 677, in start\n>>>
self.io_loop.start()\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/tornado/platform/asyncio.py\", line 199,
in start\n>>>     self.asyncio_loop.run_forever()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/asyncio/base_events.py\", line
601, in run_forever\n>>>     self._run_once()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/asyncio/base_events.py\", line
1905, in _run_once\n>>>     handle._run()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/asyncio/events.py\", line 80,
in _run\n>>>     self._context.run(self._callback, *self._args)\n>>>
\n>>>   File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/kernelbase.py\", line 457, in dispatch_queue\n>>>
await self.process_one()\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/ipykernel/kernelbase.py\", line 446, in
process_one\n>>>     await dispatch(*args)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/kernelbase.py\", line 353, in dispatch_shell\n>>>
await result\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/ipykernel/kernelbase.py\", line 648, in
execute_request\n>>>     reply_content = await reply_content\n>>> \n>>>
File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/ipkernel.py\", line 353, in do_execute\n>>>     res =
shell.run_cell(code, store_history=store_history, silent=silent)\n>>>
\n>>>   File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/zmqshell.py\", line 533, in run_cell\n>>>     return
super(ZMQInteractiveShell, self).run_cell(*args, **kwargs)\n>>> \n>>>
File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/IPython/core/interactiveshell.py\", line 2914, in run_cell\n>>>
result = self._run_cell(\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/IPython/core/interactiveshell.py\", line
2960, in _run_cell\n>>>     return runner(coro)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
```

```
packages/IPython/core/async_helpers.py\", line 78, in
_pseudo_sync_runner\n>>>     coro.send(None)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/IPython/core/interactiveshell.py\", line 3185, in
run_cell_async\n>>>     has_raised = await
self.run_ast_nodes(code_ast.body, cell_name,\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/IPython/core/interactiveshell.py\", line 3377, in
run_ast_nodes\n>>>     if (await self.run_code(code, result,
async_=asy)):\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/IPython/core/interactiveshell.py\", line
3457, in run_code\n>>>     exec(code_obj, self.user_global_ns,
self.user_ns)\n>>> \n>>>   File
\"/tmp/wsuser/ipykernel_164/3808038373.py\", line 3, in <module>\n>>>
model.fit_generator(x_train,steps_per_epoch=630,epochs=1,validation_data=
x_test,validation_steps=90)\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 1966, in
fit_generator\n>>>     return self.fit(\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 1189, in
fit\n>>>     tmp_logs = self.train_function(iterator)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 859, in
train_function\n>>>     return step_function(self, iterator)\n>>> \n>>>
File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 849, in
step_function\n>>>     outputs = model.distribute_strategy.run(run_step,
args=(data,))\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 842, in
run_step\n>>>     outputs = model.train_step(data)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 800, in
train_step\n>>>     loss = self.compiled_loss(\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/compile_utils.py\", line 204, in
__call__\n>>>     loss_value = loss_obj(y_t, y_p, sample_weight=sw)\n>>>
\n>>>   File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/losses.py\", line 155, in __call__\n>>>
losses = call_fn(y_true, y_pred)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/losses.py\", line 259, in call\n>>>
return ag_fn(y_true, y_pred, **self._fn_kwargs)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/losses.py\", line 1679, in
categorical_crossentropy\n>>>     return
backend.categorical_crossentropy(\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/backend.py\", line 4889, in
categorical_crossentropy\n>>>     return
nn.softmax_cross_entropy_with_logits_v2(\n>>> ",
     "output_type": "error",
     "traceback": [
      "\u001b[0;31m----------------------------------------------------------
---------------------\u001b[0m",
      "\u001b[0;31mInvalidArgumentError\u001b[0m
Traceback (most recent call last)",
      "\u001b[0;32m/tmp/wsuser/ipykernel_164/1479672656.py\u001b[0m in
\u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[1;32m      1\u001b[0m
\u001b[0;31m#model.fit_generator(x_train,steps_per_epoch=len(x_train),val
idation_data=x_test,validation_steps=len(x_test),epochs=10)\u001b[0m\u001
b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
2\u001b[0m \u001b[0;31m# Fitting the Model
Generator\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u
001b[0;32m----> 3\u001b[0;31m
\u001b[0mmodel\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mfit_generator\u001b
[0m\u001b[0;34m(\u001b[0m\u001b[0mx_train\u001b[0m\u001b[0;34m,\u001b[0m\
u001b[0msteps_per_epoch\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;36m630\u00
1b[0m\u001b[0;34m,\u001b[0m\u001b[0mepochs\u001b[0m\u001b[0;34m=\u001b[0m
\u001b[0;36m1\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0mvalidation_data\u001
b[0m\u001b[0;34m=\u001b[0m\u001b[0mx_test\u001b[0m\u001b[0;34m,\u001b[0m\
u001b[0mvalidation_steps\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;36m90\u00
1b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u00
1b[0m\n\u001b[0m\u001b[1;32m      4\u001b[0m
\u001b[0;31m#model.fit(x_train, epochs=100,
verbose=1)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n"
,
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\u001b[0m in
\u001b[0;36mfit_generator\u001b[0;34m(self, generator, steps_per_epoch,
epochs, verbose, callbacks, validation_data, validation_steps,
validation_freq, class_weight, max_queue_size, workers,
use_multiprocessing, shuffle, initial_epoch)\u001b[0m\n\u001b[1;32m
1964\u001b[0m                    \u001b[0;34m'will be removed in a future
version.
'\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;3
2m   1965\u001b[0m                    'Please use `Model.fit`, which
supports generators.')\n\u001b[0;32m-> 1966\u001b[0;31m     return
self.fit(\n\u001b[0m\u001b[1;32m   1967\u001b[0m
\u001b[0mgenerator\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0;34m\u001b[0m\u0
01b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m   1968\u001b[0m
\u001b[0msteps_per_epoch\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0msteps_per
_epoch\u001b[0m\u001b[0;34m,\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u0
01b[0m\u001b[0m\n",
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\u001b[0m in
\u001b[0;36mfit\u001b[0;34m(self, x, y, batch_size, epochs, verbose,
callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps,
validation_batch_size, validation_freq, max_queue_size, workers,
use_multiprocessing)\u001b[0m\n\u001b[1;32m   1187\u001b[0m
_r=1):\n\u001b[1;32m   1188\u001b[0m
\u001b[0mcallbacks\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mon_train_batch_
begin\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mstep\u001b[0m\u001b[0;34m)\u
001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m-
> 1189\u001b[0;31m                  \u001b[0mtmp_logs\u001b[0m
\u001b[0;34m=\u001b[0m
\u001b[0mself\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mtrain_function\u001b
[0m\u001b[0;34m(\u001b[0m\u001b[0miterator\u001b[0m\u001b[0;34m)\u001b[0m
\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32
m   1190\u001b[0m                  \u001b[0;32mif\u001b[0m
\u001b[0mdata_handler\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mshould_sync\
u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\
u001b[0m\n\u001b[1;32m   1191\u001b[0m
\u001b[0mcontext\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0masync_wait\u001b[
0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b
[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/util/traceback_utils.py\u001b[0m in
\u001b[0;36merror_handler\u001b[0;34m(*args,
**kwargs)\u001b[0m\n\u001b[1;32m    151\u001b[0m
\u001b[0;32mexcept\u001b[0m \u001b[0mException\u001b[0m
\u001b[0;32mas\u001b[0m
\u001b[0me\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34
m\u001b[0m\u001b[0m\n\u001b[1;32m    152\u001b[0m
\u001b[0mfiltered_tb\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0m_process_traceback_frames\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0
me\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m__traceback__\u001b[0m\u001b[0;
34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[
0;32m--> 153\u001b[0;31m       \u001b[0;32mraise\u001b[0m
\u001b[0me\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mwith_traceback\u001b[0m
\u001b[0;34m(\u001b[0m\u001b[0mfiltered_tb\u001b[0m\u001b[0;34m)\u001b[0m
\u001b[0;32mfrom\u001b[0m
\u001b[0;32mNone\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b
[0m\n\u001b[0m\u001b[1;32m    154\u001b[0m
\u001b[0;32mfinally\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u
001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m    155\u001b[0m
\u001b[0;32mdel\u001b[0m
\u001b[0mfiltered_tb\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u
001b[0m\n",
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/eager/execute.py\u001b[0m in
\u001b[0;36mquick_execute\u001b[0;34m(op_name, num_outputs, inputs,
attrs, ctx, name)\u001b[0m\n\u001b[1;32m     56\u001b[0m
\u001b[0;32mtry\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b
[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m     57\u001b[0m
\u001b[0mctx\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mensure_initialized\u0
01b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u
001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m---> 58\u001b[0;31m     tensors
= pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,
op_name,\n\u001b[0m\u001b[1;32m     59\u001b[0m
inputs, attrs, num_outputs)\n\u001b[1;32m     60\u001b[0m
\u001b[0;32mexcept\u001b[0m
\u001b[0mcore\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0m_NotOkStatusExceptio
n\u001b[0m \u001b[0;32mas\u001b[0m
\u001b[0me\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34
m\u001b[0m\u001b[0m\n",
      "\u001b[0;31mInvalidArgumentError\u001b[0m:  logits and labels must
be broadcastable: logits_size=[25,5] labels_size=[25,9]\n\t [[node
categorical_crossentropy/softmax_cross_entropy_with_logits\n (defined at
/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/backend.py:4889)\n]]
[Op:__inference_train_function_2383]\n\nErrors may have originated from
an input operation.\nInput Source operations connected to node
categorical_crossentropy/softmax_cross_entropy_with_logits:\nIn[0]
categorical_crossentropy/softmax_cross_entropy_with_logits/Reshape:\t\nIn
[1]
categorical_crossentropy/softmax_cross_entropy_with_logits/Reshape_1:\n\n
Operation defined at: (most recent call last)\n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/runpy.py\", line 197, in
_run_module_as_main\n>>>     return _run_code(code, main_globals,
None,\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/runpy.py\", line 87, in _run_code\n>>>     exec(code,
run_globals)\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/ipykernel/__main__.py\", line 3, in
<module>\n>>>     app.launch_new_instance()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/traitlets/config/application.py\", line 846, in
launch_instance\n>>>     app.start()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/kernelapp.py\", line 677, in start\n>>>
self.io_loop.start()\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/tornado/platform/asyncio.py\", line 199,
in start\n>>>     self.asyncio_loop.run_forever()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/asyncio/base_events.py\", line
601, in run_forever\n>>>     self._run_once()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/asyncio/base_events.py\", line
1905, in _run_once\n>>>     handle._run()\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/asyncio/events.py\", line 80,
in _run\n>>>     self._context.run(self._callback, *self._args)\n>>>
\n>>>   File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/kernelbase.py\", line 457, in dispatch_queue\n>>>
await self.process_one()\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/ipykernel/kernelbase.py\", line 446, in
process_one\n>>>     await dispatch(*args)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/kernelbase.py\", line 353, in dispatch_shell\n>>>
await result\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/ipykernel/kernelbase.py\", line 648, in
execute_request\n>>>     reply_content = await reply_content\n>>> \n>>>
File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/ipkernel.py\", line 353, in do_execute\n>>>     res =
shell.run_cell(code, store_history=store_history, silent=silent)\n>>>
\n>>>   File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/ipykernel/zmqshell.py\", line 533, in run_cell\n>>>     return
super(ZMQInteractiveShell, self).run_cell(*args, **kwargs)\n>>> \n>>>
File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/IPython/core/interactiveshell.py\", line 2914, in run_cell\n>>>
result = self._run_cell(\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/IPython/core/interactiveshell.py\", line
2960, in _run_cell\n>>>     return runner(coro)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/IPython/core/async_helpers.py\", line 78, in
_pseudo_sync_runner\n>>>     coro.send(None)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/IPython/core/interactiveshell.py\", line 3185, in
run_cell_async\n>>>     has_raised = await
self.run_ast_nodes(code_ast.body, cell_name,\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/IPython/core/interactiveshell.py\", line 3377, in
run_ast_nodes\n>>>     if (await self.run_code(code, result,
async_=asy)):\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-packages/IPython/core/interactiveshell.py\", line
3457, in run_code\n>>>     exec(code_obj, self.user_global_ns,
self.user_ns)\n>>> \n>>>   File
\"/tmp/wsuser/ipykernel_164/3808038373.py\", line 3, in <module>\n>>>
model.fit_generator(x_train,steps_per_epoch=630,epochs=1,validation_data=
x_test,validation_steps=90)\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 1966, in
fit_generator\n>>>     return self.fit(\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 1189, in
fit\n>>>     tmp_logs = self.train_function(iterator)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 859, in
train_function\n>>>     return step_function(self, iterator)\n>>> \n>>>
File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 849, in
step_function\n>>>     outputs = model.distribute_strategy.run(run_step,
args=(data,))\n>>> \n>>>   File \"/opt/conda/envs/Python-
3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 842, in
run_step\n>>>     outputs = model.train_step(data)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/training.py\", line 800, in
train_step\n>>>     loss = self.compiled_loss(\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/engine/compile_utils.py\", line 204, in
__call__\n>>>     loss_value = loss_obj(y_t, y_p, sample_weight=sw)\n>>>
\n>>>   File \"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/losses.py\", line 155, in __call__\n>>>
losses = call_fn(y_true, y_pred)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/losses.py\", line 259, in call\n>>>
return ag_fn(y_true, y_pred, **self._fn_kwargs)\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/losses.py\", line 1679, in
categorical_crossentropy\n>>>     return
backend.categorical_crossentropy(\n>>> \n>>>   File
\"/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/backend.py\", line 4889, in
categorical_crossentropy\n>>>     return
nn.softmax_cross_entropy_with_logits_v2(\n>>> "
     ]
    }
   ],
   "source": [

"#model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_dat
a=x_test,validation_steps=len(x_test),epochs=10)\n",
    "# Fitting the Model Generator\n",

"model.fit_generator(x_train,steps_per_epoch=630,epochs=1,validation_data
=x_test,validation_steps=90)\n",
    "#model.fit(x_train, epochs=100, verbose=1)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "Av9oIJI2P-lD"
   },
   "source": [
    "# 12.]SAVING THE MODEL"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 126,
   "metadata": {
    "id": "XEvO9YPmP08B"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "\u001b[0m\u001b[01;34mDataset\u001b[0m/
\u001b[01;34mtest_set\u001b[0m/  \u001b[01;34mtraining_set\u001b[0m/\r\n"
     ]
    }
   ],
   "source": [
    "ls"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 127,
   "metadata": {
    "id": "T1BOK_jHQIIF"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'/home/wsuser/work/Dataset'"
      ]
     },
     "execution_count": 127,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "pwd"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 128,
   "metadata": {
    "id": "OEcGdexzQL51"
   },
   "outputs": [],
   "source": [
    "model.save('Dataset.h5')"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 13.]CONVERTING ZIP FILE TO TAR FILE FOR LOCAL USE."
   ]
  },
{
 "cell_type": "code",
 "execution_count": 134,
 "metadata": {
  "scrolled": true
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Dataset.h5\r\n"
   ]
  }
 ],
 "source": [
  "#converting the model to tar\n",
  "!tar -zcvf image.Classification.model_new.tgz Dataset.h5"
 ]
},
{
 "cell_type": "code",
 "execution_count": 135,
 "metadata": {
  "scrolled": true
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "\u001b[0m\u001b[01;34mDataset\u001b[0m/\r\n",
    "Dataset.h5\r\n",
    "image.Classification.model_new.tgz\r\n",
    "\u001b[01;34mtest_set\u001b[0m/\r\n",
    "\u001b[01;34mtraining_set\u001b[0m/\r\n"
   ]
  }
 ],
 "source": [
  "ls -1"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# 14.]INSTALLING WATSON MACHINE LEARNING CLIENT SOFTWARE"
 ]
},
{
 "cell_type": "code",
 "execution_count": 137,
 "metadata": {
  "scrolled": false
 },
 "outputs": [
  {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Collecting watson_machine_learning_client\n",
       "  Downloading watson_machine_learning_client-1.0.391-py3-none-
any.whl (538 kB)\n",
       "\u001b[K     |████████████████████████████████| 538 kB 23.9 MB/s
eta 0:00:01\n",
       "\u001b[?25hRequirement already satisfied: pandas in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
watson_machine_learning_client) (1.3.4)\n",
       "Requirement already satisfied: lomond in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson_machine_learning_client)
(0.3.3)\n",
       "Requirement already satisfied: urllib3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson_machine_learning_client)
(1.26.7)\n",
       "Requirement already satisfied: requests in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson_machine_learning_client)
(2.26.0)\n",
       "Requirement already satisfied: certifi in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson_machine_learning_client)
(2022.9.24)\n",
       "Requirement already satisfied: tqdm in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson_machine_learning_client)
(4.62.3)\n",
       "Requirement already satisfied: boto3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson_machine_learning_client)
(1.18.21)\n",
       "Requirement already satisfied: tabulate in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson_machine_learning_client)
(0.8.9)\n",
       "Requirement already satisfied: ibm-cos-sdk in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
watson_machine_learning_client) (2.11.0)\n",
       "Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3-
>watson_machine_learning_client) (0.5.0)\n",
       "Requirement already satisfied: botocore<1.22.0,>=1.21.21 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3-
>watson_machine_learning_client) (1.21.41)\n",
       "Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3-
>watson_machine_learning_client) (0.10.0)\n",
       "Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
botocore<1.22.0,>=1.21.21->boto3->watson_machine_learning_client)
(2.8.2)\n",
       "Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1-
>botocore<1.22.0,>=1.21.21->boto3->watson_machine_learning_client)
(1.15.0)\n",
       "Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-
>watson_machine_learning_client) (2.11.0)\n",
       "Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-
>watson_machine_learning_client) (2.11.0)\n",
      "Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests-
>watson_machine_learning_client) (2.0.4)\n",
      "Requirement already satisfied: idna<4,>=2.5 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests-
>watson_machine_learning_client) (3.3)\n",
      "Requirement already satisfied: pytz>=2017.3 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas-
>watson_machine_learning_client) (2021.3)\n",
      "Requirement already satisfied: numpy>=1.17.3 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas-
>watson_machine_learning_client) (1.20.3)\n",
      "Installing collected packages: watson-machine-learning-client\n",
      "Successfully installed watson-machine-learning-client-1.0.391\n"
     ]
    }
   ],
   "source": [
    "#installing the machine learning repository\n",
    "!pip install watson_machine_learning_client --upgrade"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 15.]IMPORTING APICLIENT FOR DEPLOYING."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 138,
   "metadata": {},
   "outputs": [],
   "source": [
    "from ibm_watson_machine_learning import APIClient\n",
    "url_credentials = {\n",
    "    \"url\": \"https://us-south.ml.cloud.ibm.com\",\n",
    "    \"apikey\": \"sqLVTXSP3nnAKfzJ1rKRKCpNzS_XZ8_HXa9FRwV7BvOP\"\n",
    "}\n",
    "client = APIClient(url_credentials)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 139,
   "metadata": {},
   "outputs": [],
   "source": [
    "client = APIClient(url_credentials)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 16.]CREATING API_CLIENT SPACE ID. "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 140,
   "metadata": {},
   "outputs": [],
   "source": [
    "def guid_from_space_name(client, space_name):\n",
    "    space = client.spaces.get_details()\n",
    "    return(next(item for item in space['resources'] if
item['entity']['name'] == space_name)['metadata']['id'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 143,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "space UID = d90f421e-9169-47e7-a58c-0e7bb0e65685\n"
     ]
    }
   ],
   "source": [
    "space_uid = guid_from_space_name(client, 'Image Classification')\n",
    "print(\"space UID = \" + space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 145,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'SUCCESS'"
      ]
     },
     "execution_count": 145,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "client.set.default_space(space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 147,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "------------------------------  ------------------------------------
-  ----\n",
      "NAME                            ASSET_ID
TYPE\n",
      "default_py3.6                   0062b8c9-8b7d-44a0-a9b9-
46c416adcbd9  base\n",
      "kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-
31189867356a  base\n",
      "pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-
49120e15d288  base\n",
      "scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-
eb7b665ff687  base\n",
      "spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-
1ef348aebdee  base\n",
      "pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-
b5f6fccc6471  base\n",
      "ai-function_0.1-py3.6           0cdb0f1e-5376-4f4d-92dd-
da3b69aa9bda  base\n",
      "shiny-r3.6                      0e6e79df-875e-4f24-8ae9-
62dcc2148306  base\n",
      "tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-
4eb7d64b3f22  base\n",
      "pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-
3e922c096a92  base\n",
      "tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-
bf776828c4b7  base\n",
      "autoai-kb_rt22.2-py3.10         125b6d9a-5b1f-5e8d-972a-
b251688ccf40  base\n",
      "runtime-22.1-py3.9              12b83a17-24d8-5082-900f-
0ab31fbfd3cb  base\n",
      "scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-
4d5ee5abbc85  base\n",
      "default_r3.6                    1b70aec3-ab34-4b87-8aa0-
a4a3c8296a36  base\n",
      "pytorch-onnx_1.3-py3.6          1bc6029a-cc97-56da-b8e0-
39c3880dbbe7  base\n",
      "kernel-spark3.3-r3.6            1c9e5454-f216-59dd-a20e-
474a5cdf5988  base\n",
      "pytorch-onnx_rt22.1-py3.9-edt   1d362186-7ad5-5b59-8b6c-
9d0880bde37f  base\n",
      "tensorflow_2.1-py3.6            1eb25b84-d6ed-5dde-b6a5-
3fbdf1665666  base\n",
      "spark-mllib_3.2                 20047f72-0a98-58c7-9ff5-
a77b012eb8f5  base\n",
      "tensorflow_2.4-py3.8-horovod    217c16f6-178f-56bf-824a-
b19f20564c49  base\n",
      "runtime-22.1-py3.9-cuda         26215f05-08c3-5a41-a1b0-
da66306ce658  base\n",
      "do_py3.8                        295addb5-9ef9-547e-9bf4-
92ae3563e720  base\n",
      "autoai-ts_3.8-py3.8             2aa0c932-798f-5ae9-abd6-
15e0c2402fb5  base\n",
      "tensorflow_1.15-py3.6           2b73a275-7cbf-420b-a912-
eae7f436e0bc  base\n",
      "kernel-spark3.3-py3.9           2b7961e2-e3b1-5a8c-a491-
482c8368839a  base\n",
      "pytorch_1.2-py3.6              2c8ef57d-2687-4b7d-acce-
01f94976dac1  base\n",
      "spark-mllib_2.3                2e51f700-bca0-4b0d-88dc-
5c6791338875  base\n",
      "pytorch-onnx_1.1-py3.6-edt     32983cea-3f32-4400-8965-
dde874a8d67e  base\n",
      "spark-mllib_3.0-py37           36507ebe-8770-55ba-ab2a-
eafe787600e9  base\n",
      "spark-mllib_2.4                390d21f8-e58b-4fac-9c55-
d7ceda621326  base\n",
      "autoai-ts_rt22.2-py3.10        396b2e83-0953-5b86-9a55-
7ce1628a406f  base\n",
      "xgboost_0.82-py3.6             39e31acd-5f30-41dc-ae44-
60233c80306e  base\n",
      "pytorch-onnx_1.2-py3.6-edt     40589d0e-7019-4e28-8daa-
fb03b6f4fe12  base\n",
      "pytorch-onnx_rt22.2-py3.10     40e73f55-783a-5535-b3fa-
0c8b94291431  base\n",
      "default_r36py38                41c247d3-45f8-5a71-b065-
8580229facf0  base\n",
      "autoai-ts_rt22.1-py3.9         4269d26e-07ba-5d40-8f66-
2d495b0c71f7  base\n",
      "autoai-obm_3.0                 42b92e18-d9ab-567f-988a-
4240ba1ed5f7  base\n",
      "pmml-3.0_4.3                   493bcb95-16f1-5bc5-bee8-
81b8af80e9c7  base\n",
      "spark-mllib_2.4-r_3.6          49403dff-92e9-4c87-a3d7-
a42d0021c095  base\n",
      "xgboost_0.90-py3.6             4ff8d6c2-1343-4c18-85e1-
689c965304d3  base\n",
      "pytorch-onnx_1.1-py3.6         50f95b2a-bc16-43bb-bc94-
b0bed208c60b  base\n",
      "autoai-ts_3.9-py3.8            52c57136-80fa-572e-8728-
a5e7cbb42cde  base\n",
      "spark-mllib_2.4-scala_2.11     55a70f99-7320-4be5-9fb9-
9edb5a443af5  base\n",
      "spark-mllib_3.0                5c1b0ca2-4977-5c2e-9439-
ffd44ea8ffe9  base\n",
      "autoai-obm_2.0                 5c2e37fa-80b8-5e77-840f-
d912469614ee  base\n",
      "spss-modeler_18.1              5c3cad7e-507f-4b2a-a9a3-
ab53a21dee8b  base\n",
      "cuda-py3.8                     5d3232bf-c86b-5df4-a2cd-
7bb870a1cd4e  base\n",
      "autoai-kb_3.1-py3.7            632d4b22-10aa-5180-88f0-
f52dfb6444d7  base\n",
      "pytorch-onnx_1.7-py3.8         634d3cdc-b562-5bf9-a2d4-
ea90a478456b  base\n",
      "-----------------------------  -----------------------------------
-  ----\n",
      "Note: Only first 50 records were displayed. To display more use
'limit' parameter.\n"
     ]
    }
   ],
   "source": [
    "client.software_specifications.list()"
   ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "software_spec_uid =
client.software_specifications.get_uid_by_name(\"tensorflow\")\n",
    "software_spec_uid"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 17.]STORING THE MODEL_ID FOR DATASET.H5"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "#store the model\n",
    "model_details = client.repository.store_model(model='Image-
classification-model_new.tgz',meta_props={\n",
    "    client.repository.ModelMetaNames.NAME:'CNN',\n",
    "    client.repository.ModelMetaNames.TYPE:\"keras_2.2.4\",\n",
    "
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid}\n",
    "                                             )\n",
    "model_id = client.repository.get_model_uid(model_details)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "model_id"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 171,
   "metadata": {},
   "outputs": [],
   "source": [
    "model.save('Dataset.h5')"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
   "# 18.]DOWNLOADING THE TAR FILE ON CLIENT REPOSITORY"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": [
   "client.repository.download(model_id, 'my_model.tar.gz')"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "9T68YyFGQvZH"
  },
  "source": [
   "# 19.]TEST THE MODEL"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 186,
  "metadata": {
   "id": "_HAKckWyQu5C"
  },
  "outputs": [],
  "source": [
   "import numpy as np\n",
   "from tensorflow.keras.models import load_model\n",
   "from keras.preprocessing import image"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "# 20.]LOADING THE DATASET"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 187,
  "metadata": {
   "id": "69LLKetXRCPW"
  },
  "outputs": [],
  "source": [
   "#Load the model\n",
   "model=load_model('Dataset.h5')"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "# 21.]ADDING STREAMING_BODY FOR TEST IMAGE."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 188,
   "metadata": {},
   "outputs": [],
   "source": [
    "import os, types\n",
    "import pandas as pd\n",
    "from botocore.client import Config\n",
    "import ibm_boto3\n",
    "\n",
    "def __iter__(self): return 0\n",
    "\n",
    "# @hidden_cell\n",
    "# The following code accesses a file in your IBM Cloud Object
Storage. It includes your credentials.\n",
    "# You might want to remove those credentials before you share the
notebook.\n",
    "cos_client = ibm_boto3.client(service_name='s3',\n",
    "    ibm_api_key_id='aqprHZFuH38ECUn869hHk4qyvS_iKJfrZAWUJJQ-
mQKx',\n",
    "    ibm_auth_endpoint=\"https://iam.cloud.ibm.com/oidc/token\",\n",
    "    config=Config(signature_version='oauth'),\n",
    "    endpoint_url='https://s3.private.us.cloud-object-
storage.appdomain.cloud')\n",
    "\n",
    "bucket = 'realtimecommunicationforspecially-donotdelete-pr-
rfqndcvwgch6fu'\n",
    "object_key = '1.png'\n",
    "\n",
    "streaming_body_5 = cos_client.get_object(Bucket=bucket,
Key=object_key)['Body']\n",
    "\n",
    "# Your data file was loaded into a botocore.response.StreamingBody
object.\n",
    "# Please read the documentation of ibm_boto3 and pandas to learn
more about the possibilities to load the data.\n",
    "# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-
python/\n",
    "# pandas documentation: http://pandas.pydata.org/\n",
    "\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# 22.]TESTING ON SEVERAL TESTING IMAGES"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 189,
   "metadata": {},
   "outputs": [
    {
     "ename": "TypeError",
     "evalue": "expected str, bytes or os.PathLike object, not
StreamingBody",
     "output_type": "error",
     "traceback": [
      "\u001b[0;31m---------------------------------------------------------
---------------------\u001b[0m",
      "\u001b[0;31mTypeError\u001b[0m
Traceback (most recent call last)",
      "\u001b[0;32m/tmp/wsuser/ipykernel_164/365554034.py\u001b[0m in
\u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[0;32m---->
1\u001b[0;31m \u001b[0mimg\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0mimage\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mload_img\u001b[0m\u
001b[0;34m(\u001b[0m\u001b[0mstreaming_body_5\u001b[0m\u001b[0;34m,\u001b
[0m\u001b[0mtarget_size\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0;34m(\u001b
[0m\u001b[0;36m64\u001b[0m\u001b[0;34m,\u001b[0m
\u001b[0;36m64\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m)\u001b[0m\u001b
[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m
2\u001b[0m
\u001b[0;31m#img=image.load_img(\"/home/wsuser/work/1\",target_size=(64,6
4))\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/keras/preprocessing/image.py\u001b[0m in
\u001b[0;36mload_img\u001b[0;34m(path, grayscale, color_mode,
target_size, interpolation)\u001b[0m\n\u001b[1;32m    311\u001b[0m
\u001b[0;31m`\u001b[0m\u001b[0;31m`\u001b[0m\u001b[0;31m`\u001b[0m\u001b[
0mpython\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u0
01b[1;32m    312\u001b[0m
\u001b[0;34m(\u001b[0m\u001b[0mx_train\u001b[0m\u001b[0;34m,\u001b[0m
\u001b[0my_train\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m,\u001b[0m
\u001b[0;34m(\u001b[0m\u001b[0mx_test\u001b[0m\u001b[0;34m,\u001b[0m
\u001b[0my_test\u001b[0m\u001b[0;34m)\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0mcifar10\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mload_data\u001b[0
m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[
0;34m\u001b[0m\u001b[0m\n\u001b[0;32m--> 313\u001b[0;31m
\u001b[0my_train\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0mnp_utils\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mto_categorical\u
001b[0m\u001b[0;34m(\u001b[0m\u001b[0my_train\u001b[0m\u001b[0;34m,\u001b
[0m
\u001b[0mnum_classes\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\
u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m    314\u001b[0m
\u001b[0my_test\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0mnp_utils\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mto_categorical\u
001b[0m\u001b[0;34m(\u001b[0m\u001b[0my_test\u001b[0m\u001b[0;34m,\u001b[
0m
\u001b[0mnum_classes\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\
u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m    315\u001b[0m
\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/keras_preprocessing/image/utils.py\u001b[0m in
\u001b[0;36mload_img\u001b[0;34m(path, grayscale, color_mode,
target_size, interpolation)\u001b[0m\n\u001b[1;32m    111\u001b[0m
raise ImportError('Could not import PIL.Image. '\n\u001b[1;32m
112\u001b[0m                           'The use of `load_img` requires
PIL.')\n\u001b[0;32m--> 113\u001b[0;31m     \u001b[0;32mwith\u001b[0m
\u001b[0mopen\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mpath\u001b[0m\u001b[
0;34m,\u001b[0m \u001b[0;34m'rb'\u001b[0m\u001b[0;34m)\u001b[0m
\u001b[0;32mas\u001b[0m
```

```
\u001b[0mf\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34
m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m    114\u001b[0m
\u001b[0mimg\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0mpil_image\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mopen\u001b[0m\u
001b[0;34m(\u001b[0m\u001b[0mio\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mBy
tesIO\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mf\u001b[0m\u001b[0;34m.\u001
b[0m\u001b[0mread\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u0
01b[0;34m)\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34
m\u001b[0m\u001b[0m\n\u001b[1;32m    115\u001b[0m
\u001b[0;32mif\u001b[0m \u001b[0mcolor_mode\u001b[0m
\u001b[0;34m==\u001b[0m
\u001b[0;34m'grayscale'\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[
0m\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;31mTypeError\u001b[0m: expected str, bytes or os.PathLike
object, not StreamingBody"
     ]
    }
   ],
   "source": [
    "img = image.load_img(streaming_body_5,target_size=(64, 64))\n",
    "#img=image.load_img(\"/home/wsuser/work/1\",target_size=(64,64))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "IlQxA5dJRB3Q"
   },
   "outputs": [],
   "source": [
    "ls"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 181,
   "metadata": {
    "id": "kAWQrtBwRBym"
   },
   "outputs": [
    {
     "ename": "FileNotFoundError",
     "evalue": "[Errno 2] No such file or directory:
'/content/drive/MyDrive/IBM_PROJECT/Dataset/training_set/A/1.png'",
     "output_type": "error",
     "traceback": [
      "\u001b[0;31m----------------------------------------------------
--------------------\u001b[0m",
      "\u001b[0;31mFileNotFoundError\u001b[0m
Traceback (most recent call last)",
      "\u001b[0;32m/tmp/wsuser/ipykernel_164/1035932264.py\u001b[0m in
\u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[0;32m---->
1\u001b[0;31m
\u001b[0mimg\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0mimage\u001b[0m\u001b[
0;34m.\u001b[0m\u001b[0mload_img\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;3
4mr\"/content/drive/MyDrive/IBM_PROJECT/Dataset/training_set/A/1.png\"\u0
```

```
01b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34m\u001b[0m\u0
01b[0m\n\u001b[0m",
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/tensorflow/python/keras/preprocessing/image.py\u001b[0m in
\u001b[0;36mload_img\u001b[0;34m(path, grayscale, color_mode,
target_size, interpolation)\u001b[0m\n\u001b[1;32m    311\u001b[0m
\u001b[0mValueError\u001b[0m\u001b[0;34m:\u001b[0m
\u001b[0;32mif\u001b[0m \u001b[0minterpolation\u001b[0m
\u001b[0mmethod\u001b[0m \u001b[0;32mis\u001b[0m \u001b[0;32mnot\u001b[0m
\u001b[0msupported\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0;34m\u001b[0m\u0
01b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m    312\u001b[0m
\"\"\"\n\u001b[0;32m--> 313\u001b[0;31m   return image.load_img(path,
grayscale=grayscale, color_mode=color_mode,\n\u001b[0m\u001b[1;32m
314\u001b[0m                         target_size=target_size,
interpolation=interpolation)\n\u001b[1;32m    315\u001b[0m
\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;32m/opt/conda/envs/Python-3.9/lib/python3.9/site-
packages/keras_preprocessing/image/utils.py\u001b[0m in
\u001b[0;36mload_img\u001b[0;34m(path, grayscale, color_mode,
target_size, interpolation)\u001b[0m\n\u001b[1;32m    111\u001b[0m
raise ImportError('Could not import PIL.Image. '\n\u001b[1;32m
112\u001b[0m                           'The use of `load_img` requires
PIL.')\n\u001b[0;32m--> 113\u001b[0;31m     \u001b[0;32mwith\u001b[0m
\u001b[0mopen\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mpath\u001b[0m\u001b[
0;34m,\u001b[0m \u001b[0;34m'rb'\u001b[0m\u001b[0;34m)\u001b[0m
\u001b[0;32mas\u001b[0m
\u001b[0mf\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34
m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m    114\u001b[0m
\u001b[0mimg\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0mpil_image\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mopen\u001b[0m\u
001b[0;34m(\u001b[0m\u001b[0mio\u001b[0m\u001b[0;34m.\u001b[0m\u001b[0mBy
tesIO\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0mf\u001b[0m\u001b[0;34m.\u001
b[0m\u001b[0mread\u001b[0m\u001b[0;34m(\u001b[0m\u001b[0;34m)\u001b[0m\u0
01b[0;34m)\u001b[0m\u001b[0;34m)\u001b[0m\u001b[0;34m\u001b[0m\u001b[0;34
m\u001b[0m\u001b[0m\n\u001b[1;32m    115\u001b[0m
\u001b[0;32mif\u001b[0m \u001b[0mcolor_mode\u001b[0m
\u001b[0;34m==\u001b[0m
\u001b[0;34m'grayscale'\u001b[0m\u001b[0;34m:\u001b[0m\u001b[0;34m\u001b[
0m\u001b[0;34m\u001b[0m\u001b[0m\n",
      "\u001b[0;31mFileNotFoundError\u001b[0m: [Errno 2] No such file or
directory:
'/content/drive/MyDrive/IBM_PROJECT/Dataset/training_set/A/1.png'"
     ]
    }
   ],
   "source": [

"img=image.load_img(r\"/home/wsuser/work/Dataset/test_set/A/1.png\")\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "LeiulNjOSwmD"
   },
   "outputs": [],
   "source": [
    "img"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "BY0zVMqnSw--"
   },
   "outputs": [],
   "source": [

"img1=image.load_ing(r\"/home/wsuser/work/Dataset/test_set/C/1.png\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "uUXt_ZQWRBtm"
   },
   "outputs": [],
   "source": [
    "img1"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "BuWcxXfKRBie"
   },
   "outputs": [],
   "source": [
    "x=image.img_to_array(img)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "l10rMIDJRBYA"
   },
   "outputs": [],
   "source": [
    "x"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "-anXa0TFRA5O"
   },
   "outputs": [],
   "source": [
    "x1=np.expand_dims(x,axis=1)"
   ]
```

```
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "naRcte_mXUh6"
 },
 "outputs": [],
 "source": [
  "x1"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "KiRWlqpqXVLZ"
 },
 "outputs": [],
 "source": [
  "y=np.argmax(model.predoct(x),axis=1)"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "btSYV89FXVqy"
 },
 "outputs": [],
 "source": [
  "y"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "M3U9UhQFXgf1"
 },
 "outputs": [],
 "source": [
  "x_train.class_indices"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {
  "id": "KxO8yCNDXiAN"
 },
 "outputs": [],
 "source": [
  "index=['A','B','C','D','E','F','G','H','I']"
 ]
},
{
 "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "1FIK2U5oXhvO"
   },
   "outputs": [],
   "source": [
    "index[y[0]]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "NMZj0zboXhKu"
   },
   "outputs": [],
   "source": [

"img=image.load_img(r\"/home/wsuser/work/Dataset/test_set/A/90.png\",targ
et_size=(64,64))\n",
    "x=image.ing_to_array(img)\n",
    "x=np.expand_dims(x,axis=0)\n",
    "y=fnp.argmax(model.predict(x),axis=1)\n",
    "index=['A','B','C','D','E','F','G','H','I']\n",
    "index[y[0]]]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "EvUOmI7eYRn8"
   },
   "outputs": [],
   "source": [
    "img=image.load_img(
\"/home/wsuser/work/Dataset/test_set/D/1.png\",target_size=(64,64))\n",
    "x=image.ing_to_array(img)\n",
    "x=np.expand_dims(x,axis=0)\n",
    "y=np.argmax(model.predict(x)\n",
    "index=['A','B','C','D','E','F','G','H','I']\n",
    "index[y[0]]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "9HRrjTYCYRTC"
   },
   "outputs": [],
   "source": [

"img=image.load_img(r\"/content/drive/MyDrive/IBM_PROJECT/Dataset/test_se
t/G/1.png\",target_size=(64,64))\n",
    "x=image.ing_to_array(img)\n",
    "x=np.expand_dims(x,axisme)\n",
    "y=np.argmax(model.predict(x), axis=1)\n",
    "index=['A','B','C','D','E','F','G','H','I']\n",
    "index[y[0]]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "HR3o6fisYQOv"
   },
   "outputs": [],
   "source": [

"img=image.load_img(r\"/content/drive/MyDrive/IBM_PROJECT/Dataset/test_se
t/D/1.png\",target_size=(64,64))\n",
    "x-image.ing_to_array(img)\n",
    "x=np.expand_dims(x,axisme)\n",
    "y=np.argmax(model.predict(x), axis=1)\n",
    "index=['A','B','C','D','E','F','G','H','I']\n",
    "index[y[0]]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "DG_fGnHhZXJx"
   },
   "outputs": [],
   "source": [
    "!tar -zcvf Dataset-classification-model.tgz specially.h5"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "2sB_7ubnZW7p"
   },
   "outputs": [],
   "source": [
    "import tensorflow as tf\n",
    "tf .__ _version_"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "td9zCLyDb_mJ"
   },
   "outputs": [],
   "source": [
    "!pip install keras == 2.2.4"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "bUx7C1jKcRDk"
   },
   "source": [
    "# 23.]IBM DEPLOYMENT"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "0nFP_MzMcVlE"
   },
   "outputs": [],
   "source": [
    "!pip install watson-machine-learning-client  "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "I5FaOmgGca5s"
   },
   "outputs": [],
   "source": [
    "from ibm_watson_machine learning import APIClient\n",
    "wml_credentials={\n",
    "\"url\":\"https://us-south.ml.cloud.ibm.com\",\n",
    "\"apikey\":\"x91CJTUTrrIfLvrXsKf8yLyI1KHb3JV0Y7Qrwy1zilb2\"\n",
    "}\n",
    "client=APIClient(wml_credentials)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "LWiFTStydPNe"
   },
   "source": [
    "# CLIENT"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "KRfT3nwkcjqB"
   },
   "outputs": [],
   "source": [
    "def guid_space_name(client,animal_deploy):\n",
    "space-client.spaces.get_details()\n",
    "return(next(item for item in space[' resources'] if
iten['entity']['name']= animal_deploy)[\"metadata']['id'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "ToZHLNDicrmf"
   },
   "outputs": [],
   "source": [
    "space_uid-guid_space_name(client,'animal_deploy\")\n",
    "print(\"Space UID \"+space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "viITQa6edWZv"
   },
   "outputs": [],
   "source": [
    "client.set.default_space(space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "gk83aFHUdYcA"
   },
   "outputs": [],
   "source": [
    "client,software specifications.list(200)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "8_AJilmkdnFS"
   },
   "outputs": [],
   "source": [

"software_space_uid=client.software_specifications.get_uid_by_name('tenso
rflow_rt22.1-py3.9¹)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "xeAmvLnydm6h"
   },
   "outputs": [],
   "source": [
    "software_space_uid"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "tJFzLvBodmrl"
   },
   "outputs": [],
   "source": [

"model_details=client.repository.store_model(model='Dataset.tgz',meta_pro
ps={\n",
    "client.repository.ModelMetaNames.NAME: \"CNN Model Building\",\n",
    "client.repository.ModelMetaNames.TYPE: 'tensorflow_2.7',\n",
    "client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_space_uid\n",
    "})"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "ELRBCgMMdvkp"
   },
   "outputs": [],
   "source": [
    "model_id=client.repository.get_model_id(model_details)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "x1S3mF-UeqS1"
   },
   "outputs": [],
   "source": [
    "model_id"
   ]
  }
 ],
 "metadata": {
  "colab": {
   "provenance": []
  },
  "kernelspec": {
   "display_name": "Python 3.10.0 64-bit",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.10.0"
  },
  "vscode": {
   "interpreter": {
    "hash":
"26de051ba29f2982a8de78e945f0abaf191376122a1563185a90213a26c5da77"
   }
  }
 },
 "nbformat": 4,
 "nbformat_minor": 1
}
```