```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "import keras\n",
    "import tensorflow\n",
    "\n",
    "from tensorflow.keras.preprocessing.image import ImageDataGenerator"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {},
   "outputs": [],
   "source": [
    "import tensorflow\n",
    "\n",
    "from tensorflow.keras.preprocessing.image import ImageDataGenerator"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "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='JLXO-4-TMJB87CTQKc6dVclYtSXBMueJZxQKcaRUK0VP',\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 = 'forestfiredetection-donotdelete-pr-bpytmsf9pwiglr'\n",
    "object_key = 'Dataset.zip'\n",
    "\n",
    "streaming_body_1 = 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": 6,
   "metadata": {},
   "outputs": [],
   "source": [
    "from io import BytesIO\n",
    "import zipfile\n",
    "unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')\n",
    "file_paths = unzip.namelist()\n",
    "for path in file_paths:\n",
    "    unzip.extract(path)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "metadata": {},
   "outputs": [],
   "source": [
    "train_datagen = ImageDataGenerator(rescale=1./255,\n",
    "                                   shear_range=0.2,\n",
    "                                   rotation_range=180,\n",
    "                                   zoom_range=0.2,\n",
    "                                   horizontal_flip=True)\n",
    "\n",
    "test_datagen = ImageDataGenerator(rescale=1./255)"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Found 436 images belonging to 2 classes.\n"
   ]
  }
 ],
 "source": [
  "x_train = train_datagen.flow_from_directory(r'./Dataset/train_set/',\n",
  "                                            target_size=(128, 128),\n",
  "                                            batch_size=32,\n",
  "                                            class_mode='binary')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 14,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Found 121 images belonging to 2 classes.\n"
   ]
  }
 ],
 "source": [
  "x_test = train_datagen.flow_from_directory(r'./Dataset/test_set/',\n",
  "                                            target_size=(128, 128),\n",
  "                                            batch_size=32,\n",
  "                                            class_mode='binary')"
 ]
},
{
 "cell_type": "code",
 "execution_count": 15,
 "metadata": {},
```

```
   "outputs": [],
   "source": [
    "from tensorflow.keras.models import Sequential\n",
    "from tensorflow.keras.layers import Dense, Convolution2D, MaxPooling2D,
Flatten\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "metadata": {},
   "outputs": [],
   "source": [
    "model = Sequential()\n",
    "model.add(Convolution2D(32, (3,3), input_shape=(128, 128, 3),
activation=\"relu\"))\n",
    "model.add(MaxPooling2D(pool_size=(2,2)))\n",
    "model.add(Flatten())\n",
    "model.add(Dense(150,activation=\"relu\"))\n",
    "model.add(Dense(1, activation=\"sigmoid\"))\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 17,
   "metadata": {},
   "outputs": [],
   "source": [
    "model.compile(loss=\"binary_crossentropy\",\n",
    "              optimizer=\"adam\",\n",
    "              metrics=[\"accuracy\"])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Epoch 1/10\n",
      "14/14 [==============================] - 24s 2s/step - loss: 2.0141 -
accuracy: 0.7133 - val_loss: 0.1630 - val_accuracy: 0.9421\n",
      "Epoch 2/10\n",
```

```
      "14/14 [==============================] - 22s 2s/step - loss: 0.3240 -
accuracy: 0.8922 - val_loss: 0.1051 - val_accuracy: 0.9835\n",
      "Epoch 3/10\n",
      "14/14 [==============================] - 22s 2s/step - loss: 0.2306 -
accuracy: 0.9014 - val_loss: 0.1186 - val_accuracy: 0.9421\n",
      "Epoch 4/10\n",
      "14/14 [==============================] - 21s 2s/step - loss: 0.1938 -
accuracy: 0.9174 - val_loss: 0.0852 - val_accuracy: 0.9752\n",
      "Epoch 5/10\n",
      "14/14 [==============================] - 22s 1s/step - loss: 0.1953 -
accuracy: 0.9243 - val_loss: 0.1242 - val_accuracy: 0.9339\n",
      "Epoch 6/10\n",
      "14/14 [==============================] - 21s 2s/step - loss: 0.1797 -
accuracy: 0.9128 - val_loss: 0.0790 - val_accuracy: 0.9835\n",
      "Epoch 7/10\n",
      "14/14 [==============================] - 21s 1s/step - loss: 0.1688 -
accuracy: 0.9335 - val_loss: 0.0905 - val_accuracy: 0.9421\n",
      "Epoch 8/10\n",
      "14/14 [==============================] - 21s 1s/step - loss: 0.1727 -
accuracy: 0.9220 - val_loss: 0.1370 - val_accuracy: 0.9256\n",
      "Epoch 9/10\n",
      "14/14 [==============================] - 22s 2s/step - loss: 0.2078 -
accuracy: 0.9128 - val_loss: 0.0687 - val_accuracy: 0.9917\n",
      "Epoch 10/10\n",
      "14/14 [==============================] - 22s 2s/step - loss: 0.1622 -
accuracy: 0.9266 - val_loss: 0.0673 - val_accuracy: 0.9835\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "<keras.callbacks.History at 0x7f63f52b16d0>"
      ]
     },
     "execution_count": 18,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "model.fit(x_train, steps_per_epoch=14, epochs=10,
validation_data=x_test, validation_steps=4)"
   ]
  },
  {
   "cell_type": "markdown",
```

```
 "metadata": {},
 "source": [
  "Save the model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 19,
 "metadata": {},
 "outputs": [],
 "source": [
  "model.save(\"model.h5\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 20,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "model.h5\r\n"
   ]
  }
 ],
 "source": [
  "!tar -zcvf model.tgz model.h5"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "Prediction"
 ]
},
{
 "cell_type": "code",
 "execution_count": 21,
 "metadata": {},
 "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 18.1 MB/s eta
0:00:01\n",
      "\u001b[?25hRequirement 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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-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: 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: 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": [
    "!pip install watson-machine-learning-client"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "metadata": {},
   "outputs": [],
   "source": [
    "from ibm_watson_machine_learning import APIClient\n",
    "\n",
    "API_KEY = \"5W65wtnV1kus0WCtJ4HEMzw-lVetPUlY_B2Nje3fDq4p\"\n",
    "\n",
    "credentials = {\n",
    "    \"url\": \"https://us-south.ml.cloud.ibm.com\",\n",
    "    \"apikey\": API_KEY\n",
    "}\n",
    "\n",
    "client = APIClient(credentials)"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "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": 32,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "{'resources': []}"
      ]
     },
     "execution_count": 32,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    " space = client.spaces.get_details()\n",
    " space"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 33,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Space UID:  d9308ab8-179a-48da-974b-d986f1649bd5\n"
     ]
```

```
    }
   ],
   "source": [
    "space_uid = guid_from_space_name(client, 'Forest fire detection')\n",
    "print(\"Space UID: \", space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 34,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'SUCCESS'"
      ]
     },
     "execution_count": 34,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "client.set.default_space(space_uid)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 35,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'12b83a17-24d8-5082-900f-0ab31fbfd3cb'"
      ]
     },
     "execution_count": 35,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "software_spec_uid =
client.software_specifications.get_uid_by_name(\"runtime-22.1-py3.9\")\n",
```

```
    "software_spec_uid"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 36,
   "metadata": {},
   "outputs": [],
   "source": [
    "model_details = client.repository.store_model(model=\"model.tgz\",
meta_props={\n",
    "    client.repository.ModelMetaNames.NAME: \"CNN\",\n",
    "    client.repository.ModelMetaNames.TYPE: \"tensorflow_2.7\",\n",
    "    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid\n",
    "})\n",
    "\n",
    "model_id = client.repository.get_model_id(model_details)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Successfully saved model content to file: 'model.tar.gz'\n"
     ]
    },
    {
     "data": {
      "text/plain": [
       "'/home/wsuser/work/model.tar.gz'"
      ]
     },
     "execution_count": 37,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "client.repository.download(model_id, \"model.tar.gz\")"
   ]
```

```
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3.9",
   "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": "3.9.13"
  },
  "vscode": {
   "interpreter": {
    "hash":
"a9cff5a362bc38ef45d817ae74b1af54d6a076e3d773891282bce078b815ba34"
   }
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```