

SPRINT-4

MODEL DEPLOYMENT

Date	14-November-22
Team ID	PNT2022TMID15393
Project Title	A Gesture-based Tool for Sterile Browsing of Radiology Images

IBM WATSON:

```
Projects / A Gesture-Based Tool For Sterile ... / Model_building

In [14]: import os, types
import pandas as pd
from boto3.core.client import Config
import ibm_boto3

def __iter__(self): return 0

#@hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
                              ibm_api_key_id='tAs6NHrXkeZtYKGMuB4Kcz1zvK9jbBVa-3a-i1TXDHR',
                              ibm_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
                              config=Config(signature_version='oauth'),
                              endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'gesturebasedtoolforsterilebrowsi-donotdelete-pr-okekrvpncufqa2'
object_key = 'Dataset.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/

In [15]: from io import BytesIO
import zipfile
```

```
In [20]: x_train = train_datagen.flow_from_directory(r"Dataset/train/",
                                                    target_size=(64, 64),
                                                    batch_size=3,
                                                    color_mode='grayscale',
                                                    class_mode='categorical')

#performing data agumentation to test data
x_test = test_datagen.flow_from_directory(r"Dataset/test/",
                                           target_size=(64, 64),
                                           batch_size=3,
                                           color_mode='grayscale',
                                           class_mode='categorical')

Found 594 images belonging to 6 classes.
Found 30 images belonging to 6 classes.
```

```
In [21]: print(x_train.class_indices)

{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5}

Model Creation
```

```
In [22]: model = Sequential()
```

```
In [23]: # First convolution Layer and pooling
model.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
In [24]: # Second convolution Layer and pooling
model.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution Layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
In [25]: # Flattening the Layers i.e. input Layer
model.add(Flatten())
```

```
In [26]: # Adding a fully connected Layer, i.e. Hidden Layer
model.add(Dense(units=512, activation='relu'))
```

```
In [27]: # softmax for categorical analysis, Output Layer
model.add(Dense(units=6, activation='softmax'))
```

```
In [28]: model.summary()
```

```
Model: "sequential"

Layer (type)                 Output Shape                 Param #
-----
conv2d (Conv2D)              (None, 62, 62, 32)          320
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)          0
conv2d_1 (Conv2D)            (None, 29, 29, 32)          9248
max_pooling2d_1 (MaxPooling2D) (None, 14, 14, 32)          0
flatten (Flatten)            (None, 6272)                0
dense (Dense)                (None, 512)                 3211776
dense_1 (Dense)              (None, 6)                   3078
Total params: 3,224,422
Trainable params: 3,224,422
Non-trainable params: 0
```

Model Compilation

```
In [29]: # Compiling the CNN
# categorical_crossentropy for more than 2
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Model Fitting

```
In [30]: # It will generate packets of train and test data for training
model.fit_generator(x_train,
                    steps_per_epoch = 594/3,
                    epochs = 25,
                    validation_data = x_test,
                    validation_steps = 30/3)
```

```
Epoch 1/25
198/198 [=====] - 14s 68ms/step - loss: 1.2871 - accuracy: 0.5152 - val_loss: 0.7762 - val_accuracy: 0.6333
Epoch 2/25
198/198 [=====] - 13s 68ms/step - loss: 0.5824 - accuracy: 0.7559 - val_loss: 0.4198 - val_accuracy: 0.8333
Epoch 3/25
198/198 [=====] - 13s 67ms/step - loss: 0.4235 - accuracy: 0.8300 - val_loss: 0.1689 - val_accuracy: 0.9667
Epoch 4/25
198/198 [=====] - 15s 75ms/step - loss: 0.2671 - accuracy: 0.8939 - val_loss: 0.2322 - val_accuracy: 0.9667
Epoch 5/25
198/198 [=====] - 14s 69ms/step - loss: 0.2073 - accuracy: 0.9192 - val_loss: 0.2652 - val_accuracy: 0.9333
Epoch 6/25
198/198 [=====] - 14s 68ms/step - loss: 0.1325 - accuracy: 0.9562 - val_loss: 0.2813 - val_accuracy: 0.9333
Epoch 7/25
198/198 [=====] - 14s 69ms/step - loss: 0.1427 - accuracy: 0.9596 - val_loss: 0.2561 - val_accuracy: 0.9000
Epoch 8/25
198/198 [=====] - 13s 68ms/step - loss: 0.1078 - accuracy: 0.9579 - val_loss: 0.2664 - val_accuracy: 0.9667
Epoch 9/25
198/198 [=====] - 15s 76ms/step - loss: 0.1224 - accuracy: 0.9495 - val_loss: 0.1192 - val_accuracy: 0.9333
Epoch 10/25
198/198 [=====] - 14s 68ms/step - loss: 0.0755 - accuracy: 0.9731 - val_loss: 0.4680 - val_accuracy: 0.9333
Epoch 11/25
198/198 [=====] - 13s 66ms/step - loss: 0.0733 - accuracy: 0.9747 - val_loss: 0.3866 - val_accuracy: 0.9000
Epoch 12/25
198/198 [=====] - 14s 68ms/step - loss: 0.0405 - accuracy: 0.9815 - val_loss: 0.5000 - val_accuracy: 0.9333
Epoch 13/25
198/198 [=====] - 14s 73ms/step - loss: 0.0624 - accuracy: 0.9781 - val_loss: 0.3178 - val_accuracy: 0.9667
Epoch 14/25
198/198 [=====] - 13s 67ms/step - loss: 0.0678 - accuracy: 0.9798 - val_loss: 0.5067 - val_accuracy: 0.9000
Epoch 15/25
198/198 [=====] - 13s 66ms/step - loss: 0.0712 - accuracy: 0.9798 - val_loss: 0.3540 - val_accuracy: 0.9333
Epoch 16/25
198/198 [=====] - 13s 67ms/step - loss: 0.0619 - accuracy: 0.9848 - val_loss: 0.3448 - val_accuracy: 0.9667
Epoch 17/25
198/198 [=====] - 13s 68ms/step - loss: 0.0455 - accuracy: 0.9865 - val_loss: 0.2879 - val_accuracy: 0.9667
Epoch 18/25
198/198 [=====] - 14s 70ms/step - loss: 0.0679 - accuracy: 0.9815 - val_loss: 0.3151 - val_accuracy: 0.9667
Epoch 19/25
198/198 [=====] - 14s 70ms/step - loss: 0.0276 - accuracy: 0.9916 - val_loss: 0.2860 - val_accuracy: 0.9667
Epoch 20/25
198/198 [=====] - 13s 67ms/step - loss: 0.0615 - accuracy: 0.9798 - val_loss: 0.4351 - val_accuracy: 0.9667
Epoch 21/25
198/198 [=====] - 14s 70ms/step - loss: 0.0146 - accuracy: 0.9933 - val_loss: 0.3405 - val_accuracy: 0.9667
Epoch 22/25
198/198 [=====] - 14s 69ms/step - loss: 0.0086 - accuracy: 0.9983 - val_loss: 0.4429 - val_accuracy: 0.9667
Epoch 23/25
198/198 [=====] - 13s 66ms/step - loss: 0.0228 - accuracy: 0.9949 - val_loss: 0.5591 - val_accuracy: 0.9667
Epoch 24/25
198/198 [=====] - 13s 67ms/step - loss: 0.0183 - accuracy: 0.9916 - val_loss: 0.6934 - val_accuracy: 0.9000
Epoch 25/25
198/198 [=====] - 13s 67ms/step - loss: 0.0271 - accuracy: 0.9882 - val_loss: 0.4183 - val_accuracy: 0.9333
Out[30]: <keras.callbacks.History at 0x7f5524ebcbe0>
```

Saving the Model

```
In [31]: model.save('gesture.h5')

In [32]: !tar -zcvf gesture-classification-model.tgz gesture.h5
gesture.h5
```

IBM Deployment

```
In [33]: !pip install ibm_watson_machine_learning

Requirement already satisfied: ibm_watson_machine_learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.255)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2022.9.24)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (4.8.2)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (21.3)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.3.4)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.26.7)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.26.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (0.10.0)
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==2.11.*->ibm_watson_machine_learning) (2.11.0)
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==2.11.*->ibm_watson_machine_learning) (2.11.0)

In [69]: from ibm_watson_machine_learning import APIClient

In [70]: wml_credentials = {
        "url": "https://us-south.ml.cloud.ibm.com",
        "apikey": "00QFbTKSta9PaJo2rUTKC-UR1xSvPn8A_9cH60GE6XQP"
    }

In [71]: client = APIClient(wml_credentials)

In [72]: client

Out[72]: <ibm_watson_machine_learning.client.APIClient at 0x7f54acbb9340>

In [73]: client.spaces.list()

Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
-----
ID          NAME          CREATED
a76bf5bd-8553-45c8-bff9-ac13291f3e65  Gesture based tool  2022-11-01T09:34:21.631Z
-----

In [74]: space_id = "a76bf5bd-8553-45c8-bff9-ac13291f3e65"
```

```

Epoch 1/25
198/198 [=====] - 14s 68ms/step - loss: 1.2871 - accuracy: 0.5152 - val_loss: 0.7762 - val_accuracy: 0.6333
Epoch 2/25
198/198 [=====] - 13s 68ms/step - loss: 0.5824 - accuracy: 0.7559 - val_loss: 0.4198 - val_accuracy: 0.8333
Epoch 3/25
198/198 [=====] - 13s 67ms/step - loss: 0.4235 - accuracy: 0.8300 - val_loss: 0.1689 - val_accuracy: 0.9667
Epoch 4/25
198/198 [=====] - 15s 75ms/step - loss: 0.2671 - accuracy: 0.8939 - val_loss: 0.2322 - val_accuracy: 0.9667
Epoch 5/25
198/198 [=====] - 14s 69ms/step - loss: 0.2073 - accuracy: 0.9192 - val_loss: 0.2652 - val_accuracy: 0.9333
Epoch 6/25
198/198 [=====] - 14s 68ms/step - loss: 0.1325 - accuracy: 0.9562 - val_loss: 0.2813 - val_accuracy: 0.9333
Epoch 7/25
198/198 [=====] - 14s 69ms/step - loss: 0.1427 - accuracy: 0.9596 - val_loss: 0.2561 - val_accuracy: 0.9000
Epoch 8/25
198/198 [=====] - 13s 68ms/step - loss: 0.1078 - accuracy: 0.9579 - val_loss: 0.2664 - val_accuracy: 0.9667
Epoch 9/25
198/198 [=====] - 15s 76ms/step - loss: 0.1224 - accuracy: 0.9495 - val_loss: 0.1192 - val_accuracy: 0.9333
Epoch 10/25
198/198 [=====] - 14s 68ms/step - loss: 0.0755 - accuracy: 0.9731 - val_loss: 0.4680 - val_accuracy: 0.9333
Epoch 11/25
198/198 [=====] - 13s 66ms/step - loss: 0.0733 - accuracy: 0.9747 - val_loss: 0.3866 - val_accuracy: 0.9000
Epoch 12/25
198/198 [=====] - 14s 68ms/step - loss: 0.0405 - accuracy: 0.9815 - val_loss: 0.5000 - val_accuracy: 0.9333
Epoch 13/25
198/198 [=====] - 14s 73ms/step - loss: 0.0624 - accuracy: 0.9781 - val_loss: 0.3178 - val_accuracy: 0.9667
Epoch 14/25
198/198 [=====] - 13s 67ms/step - loss: 0.0678 - accuracy: 0.9798 - val_loss: 0.5067 - val_accuracy: 0.9000
Epoch 15/25
198/198 [=====] - 13s 66ms/step - loss: 0.0712 - accuracy: 0.9798 - val_loss: 0.3540 - val_accuracy: 0.9333

```

```
In [77]: software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt2.1-py3.9')
```

```
In [83]: software_space_uid
```

```
Out[83]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'
```

```
In [ ]: model_details=client.repository.store_model(model='gesture-classification-model.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"CNN Model Building",
    client.repository.ModelMetaNames.TYPE:'tensorflow_2.7',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
})
```

```
In [81]: model_id = client.repository.get_model_id(model_details)
```

```
In [82]: model_id
```

```
Out[82]: '48c7db6a-7364-4576-8944-628211cc3bd4'
```

```
In [ ]:
```

Deployments

1 space

New deployment space +

Activity

Spaces

Filter by: All spaces ▾ 🔍 Which deployment space are you looking for?

Name	Last modified ↓	Your role	Collaborators	Tags	Online deployments ⓘ	Jobs ⓘ
Gesture based tool	Nov 14, 2022, 3:04 PM	Admin	RR VS SM		0	0