Watson OpenScale *sample notebook*

# Tutorial on generating an explanation for an image-based model on Watson OpenScale

This notebook includes steps for creating an image-based watson-machine-learning model, creating a subscription, configuring explainability, and finally generating an explanation for a transaction.

## Contents

**Note**: If using Watson Studio, try running the notebook on at least 'Default Python 3.5 S' version for faster results (vs Python XS).

# 1. Setup

## 1.1 Install Watson OpenScale and WML packages

```
!pip install --upgrade ibm-ai-openscale
!pip install --upgrade watson-machine-learning-client --no-cache | tail -n 1

# !pip install --upgrade watson-machine-learning-client --no-cache | tail -n 1
# !pip install watson-machine-learning-client==1.0.371

#  !pip install watson-machine-learning-client==1.0.375
# !pip install --upgrade ibm-ai-openscale

# !pip install --upgrade ibm-ai-openscale --no-cache | tail -n 1
# !pip install ibm-ai-openscale==2.1.16

# !pip install --upgrade watson-machine-learning-client --no-cache | tail -n 1
# !pip install watson-machine-learning-client==1.0.371
```

```
Collecting ibm-ai-openscale
[?25l  Downloading https://files.pythonhosted.org/packages/31/f9/5167f4954c06351f7e65365c9af475edfab96d8f424e2c772d4c1c3c9802/ibm_ai_openscale-2.1.17-py3-none-any.whl (537kB)
[K     |████████████████████████████████| 542kB 15.6MB/s eta 0:00:01
[?25hRequirement already satisfied, skipping upgrade: pandas in /opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-ai-openscale) (0.24.1)
Requirement already satisfied, skipping upgrade: tabulate in /opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-ai-openscale) (0.8.2)
Requirement already satisfied, skipping upgrade: requests in /opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-ai-openscale) (2.21.0)
Requirement already satisfied, skipping upgrade: h5py in /opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-ai-openscale) (2.9.0)
Requirement already satisfied, skipping upgrade: pytz>=2011k in /opt/conda/envs/Python36/lib/python3.6/site-packages (from pandas->ibm-ai-openscale) (2018.9)
Requirement already satisfied, skipping upgrade: numpy>=1.12.0 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from pandas->ibm-ai-openscale) (1.15.4)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.5.0 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from pandas->ibm-ai-openscale) (2.7.5)
Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from requests->ibm-ai-openscale) (2.8)
Requirement already satisfied, skipping upgrade: urllib3<1.25,>=1.21.1 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from requests->ibm-ai-openscale) (1.24.1)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from requests->ibm-ai-openscale) (2019.9.11)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from requests->ibm-ai-openscale) (3.0.4)
Requirement already satisfied, skipping upgrade: six in /opt/conda/envs/Python36/lib/python3.6/site-packages (from h5py->ibm-ai-openscale) (1.12.0)
Installing collected packages: ibm-ai-openscale
Successfully installed ibm-ai-openscale-2.1.17
Successfully installed watson-machine-learning-client-1.0.376
```

Note: Restart the kernel to assure the new libraries are being used.

## 1.2 Configure credentials

To run this Lab you need to have a valid instance of Watson Openscale.

To verify if you have one, go to the cloud console, clicking on `Services` you should see your Watson OpenScale instance listed.

if not then from that screen click the upper right button **"Create ressource"**. From the search entry type 'openscale' and create a lite plan of Watson OpenScale.

You also need a valid **IBM Cloud API Key** to assign the variable in the next cell.

To get it go to the IBM Cloud console then click from the upper toolbar `Manage->Access (IAM)`. Select `IBM Cloud API Keys` from the left hand sidebar and then click the **"Create an IBM Cloud API Key"** button.

From that page, give your key a name and click Create, then copy the created key and paste it below.

```
CLOUD_API_KEY = "<insert your own CLOUD-API-KEY here>"
```

```
import requests
from ibm_ai_openscale.utils import get_instance_guid

WOS_GUID = get_instance_guid(api_key=CLOUD_API_KEY)
AIOS_CREDENTIALS = {
    "instance_guid": WOS_GUID,
    "apikey": CLOUD_API_KEY,
    "url": "https://api.aiopenscale.cloud.ibm.com"
}

if WOS_GUID is None:
    print('Watson OpenScale GUID NOT FOUND')
else:
    print(WOS_GUID)
```

```
70ee9046-f34e-441c-8dbe-75d57d88b6f7
```

You also need to have a valid instance of Watson Machine Learning (runtime for your models) running.

To verify if you have one, go to the cloud console, clicking on `Services` you should see your Watson Machine Learning instance listed.

if not then from that screen click the upper right button **"Create ressource"**. From the search entry type 'Machine Learning' and create a lite plan of Watson Machine Learning. MAKE SURE THE REGION FIELD GOT **DALLAS** as value if not modify it accordingly.

From the IBM Cloud Resource list click on the Watson Machine Learning instance and from this page click the service credentials side bar item. clik on view **'credentials'** and copy the all json info provided as follow :

```
{
  "apikey": "XXXXXXXXXX",
  "iam_apikey_description": "Auto-generated for key XXXX-YYYYY-ZZZZZZ",
  "iam_apikey_name": "WML-credentials",
  "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/XXXXXXXX::serviceid:
ServiceId-XXXX-YYYYYY-ZZZZZZZZ",
  "instance_id": "WWWWWWWWWWWWWWWW",
  "url": "https://us-south.ml.cloud.ibm.com"
}
```

replace the following variable with the obtained json data.

```

WML_CREDENTIALS = {
  "apikey": "xxxxxxxxxxxxxx",
  "iam_apikey_description": "Auto-generated for key yyyyyyyyyyyyy",
  "iam_apikey_name": "Service credentials-WML4JLC",
  "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Writer",zzzzz",
  "instance_id": "xxxxxxx",
  "url": "https://us-south.ml.cloud.ibm.com"
}
```

```
import sys, time

def Wait(seconds, Speed=5):
    Chars = ["|","/","-","\\"]
    MaxChars = 4
    sys.stdout.flush()
    for i in range(seconds*Speed):
        sys.stdout.write("\r" + Chars[i % MaxChars])
        sys.stdout.flush()
        time.sleep(1/Speed)
    sys.stdout.write("\r ")
Wait(10)
```

# 2. Creating and deploying an image-based model

The dataset used is MNIST dataset of handwritten digits. It consists of 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images. More information about the dataset can be found here: https://keras.io/datasets/#mnist-database-of-handwritten-digits

Note: Tensorflow versions supported by WML are: 1.2, 1.5, and 1.11. Make sure you have one of these versions before creating the models. Version 1.11 is used in this notebook.

## 2.1 Creating a model

```
!pip install keras
!pip install tensorflow==1.11.0
!pip install keras_sequential_ascii

import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras_sequential_ascii import sequential_model_to_ascii_printout
from keras import backend as keras_backend
```

```
Requirement already satisfied: keras in /opt/conda/envs/Python36/lib/python3.6/sit
e-packages (2.2.4)
Requirement already satisfied: numpy>=1.9.1 in /opt/conda/envs/Python36/lib/python
3.6/site-packages (from keras) (1.15.4)
Requirement already satisfied: scipy>=0.14 in /opt/conda/envs/Python36/lib/python3
.6/site-packages (from keras) (1.2.0)
Requirement already satisfied: six>=1.9.0 in /opt/conda/envs/Python36/lib/python3.
6/site-packages (from keras) (1.12.0)
Requirement already satisfied: pyyaml in /opt/conda/envs/Python36/lib/python3.6/si
te-packages (from keras) (3.13)
Requirement already satisfied: h5py in /opt/conda/envs/Python36/lib/python3.6/site
-packages (from keras) (2.9.0)
Requirement already satisfied: keras_applications>=1.0.6 in /opt/conda/envs/Python
36/lib/python3.6/site-packages (from keras) (1.0.6)
Requirement already satisfied: keras_preprocessing>=1.0.5 in /opt/conda/envs/Pytho
n36/lib/python3.6/site-packages (from keras) (1.0.5)
Collecting tensorflow==1.11.0
[?25l  Downloading https://files.pythonhosted.org/packages/ce/d5/38cd4543401708e64
c9ee6afa664b936860f4630dd93a49ab863f9998cd2/tensorflow-1.11.0-cp36-cp36m-manylinux
1_x86_64.whl (63.0MB)
[K     |████████████████████████████████| 63.0MB 45.6MB/s eta 0:00:01
[?25hRequirement already satisfied: keras-applications>=1.0.5 in /opt/conda/envs/P
ython36/lib/python3.6/site-packages (from tensorflow==1.11.0) (1.0.6)
Requirement already satisfied: six>=1.10.0 in /opt/conda/envs/Python36/lib/python3
.6/site-packages (from tensorflow==1.11.0) (1.12.0)
Requirement already satisfied: wheel>=0.26 in /opt/conda/envs/Python36/lib/python3
.6/site-packages (from tensorflow==1.11.0) (0.32.3)
Requirement already satisfied: keras-preprocessing>=1.0.3 in /opt/conda/envs/Pytho
n36/lib/python3.6/site-packages (from tensorflow==1.11.0) (1.0.5)
Collecting tensorboard<1.12.0,>=1.11.0 (from tensorflow==1.11.0)
[?25l  Downloading https://files.pythonhosted.org/packages/9b/2f/4d788919b1feef046
24d63ed6ea45a49d1d1c834199ec50716edb5d310f4/tensorboard-1.11.0-py3-none-any.whl (3
.0MB)
```

```
[K     |████████████████████████████████| 3.0MB 39.7MB/s eta 0:00:01:01�██████████
███▌   | 2.9MB 39.7MB/s eta 0:00:01
[?25hRequirement already satisfied: astor>=0.6.0 in /opt/conda/envs/Python36/lib/p
ython3.6/site-packages (from tensorflow==1.11.0) (0.7.1)
Requirement already satisfied: protobuf>=3.6.0 in /opt/conda/envs/Python36/lib/pyt
hon3.6/site-packages (from tensorflow==1.11.0) (3.6.1)
Collecting setuptools<=39.1.0 (from tensorflow==1.11.0)
[?25l  Downloading https://files.pythonhosted.org/packages/8c/10/79282747f9169f21c
053c562a0baa21815a8c7879be97abd930dbcf862e8/setuptools-39.1.0-py2.py3-none-any.whl
 (566kB)
[K     |████████████████████████████████| 573kB 37.4MB/s eta 0:00:01
[?25hRequirement already satisfied: grpcio>=1.8.6 in /opt/conda/envs/Python36/lib/
python3.6/site-packages (from tensorflow==1.11.0) (1.16.1)
Requirement already satisfied: absl-py>=0.1.6 in /opt/conda/envs/Python36/lib/pyth
on3.6/site-packages (from tensorflow==1.11.0) (0.7.0)
Requirement already satisfied: termcolor>=1.1.0 in /opt/conda/envs/Python36/lib/py
thon3.6/site-packages (from tensorflow==1.11.0) (1.1.0)
Requirement already satisfied: gast>=0.2.0 in /opt/conda/envs/Python36/lib/python3
.6/site-packages (from tensorflow==1.11.0) (0.2.2)
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/envs/Python36/lib/pytho
n3.6/site-packages (from tensorflow==1.11.0) (1.15.4)
Requirement already satisfied: h5py in /opt/conda/envs/Python36/lib/python3.6/site
-packages (from keras-applications>=1.0.5->tensorflow==1.11.0) (2.9.0)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/envs/Python36/lib/pyt
hon3.6/site-packages (from tensorboard<1.12.0,>=1.11.0->tensorflow==1.11.0) (3.0.1
)
Requirement already satisfied: werkzeug>=0.11.10 in /opt/conda/envs/Python36/lib/p
ython3.6/site-packages (from tensorboard<1.12.0,>=1.11.0->tensorflow==1.11.0) (0.1
4.1)
Installing collected packages: tensorboard, setuptools, tensorflow
  Found existing installation: setuptools 40.8.0
    Uninstalling setuptools-40.8.0:
      Successfully uninstalled setuptools-40.8.0
  Found existing installation: tensorflow 1.13.1
    Uninstalling tensorflow-1.13.1:
      Successfully uninstalled tensorflow-1.13.1
Successfully installed setuptools-39.1.0 tensorboard-1.11.0 tensorflow-1.11.0
Collecting keras_sequential_ascii
  Downloading https://files.pythonhosted.org/packages/2d/a4/806e3ed5d7ac7463e2fae7
7e09ccccc88c78266b248fb637e4efa4f65ec0/keras_sequential_ascii-0.1.1.tar.gz
Requirement already satisfied: keras in /opt/conda/envs/Python36/lib/python3.6/sit
e-packages (from keras_sequential_ascii) (2.2.4)
Requirement already satisfied: numpy>=1.9.1 in /opt/conda/envs/Python36/lib/python
3.6/site-packages (from keras->keras_sequential_ascii) (1.15.4)
Requirement already satisfied: scipy>=0.14 in /opt/conda/envs/Python36/lib/python3
.6/site-packages (from keras->keras_sequential_ascii) (1.2.0)
Requirement already satisfied: six>=1.9.0 in /opt/conda/envs/Python36/lib/python3.
6/site-packages (from keras->keras_sequential_ascii) (1.12.0)
```

```
Requirement already satisfied: pyyaml in /opt/conda/envs/Python36/lib/python3.6/si
te-packages (from keras->keras_sequential_ascii) (3.13)
Requirement already satisfied: h5py in /opt/conda/envs/Python36/lib/python3.6/site
-packages (from keras->keras_sequential_ascii) (2.9.0)
Requirement already satisfied: keras_applications>=1.0.6 in /opt/conda/envs/Python
36/lib/python3.6/site-packages (from keras->keras_sequential_ascii) (1.0.6)
Requirement already satisfied: keras_preprocessing>=1.0.5 in /opt/conda/envs/Pytho
n36/lib/python3.6/site-packages (from keras->keras_sequential_ascii) (1.0.5)
Building wheels for collected packages: keras-sequential-ascii
  Building wheel for keras-sequential-ascii (setup.py) ... [?25ldone
[?25h  Stored in directory: /home/dsxuser/.cache/pip/wheels/f5/8d/81/912666dff82a9
23ce423a7e797cd75f54271c7031512cdb282
Successfully built keras-sequential-ascii
Installing collected packages: keras-sequential-ascii
Successfully installed keras-sequential-ascii-0.1.1



Using TensorFlow backend.
```

```
print("KERAS v {}".format(keras.__version__))

import tensorflow as tf
print("TENSORFLOW v {}".format(tf.__version__))
```

```
KERAS v 2.2.4
TENSORFLOW v 1.11.0
```

```
!ls
```

```
batch_size = 128
num_classes = 10
epochs = 5
```

```
# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if keras_backend.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

if you don't want to train the model during this lab, which is quite time consumming (15 mn average) depending on the size of your python/jupyter environment, you can use a pre-trained model provided to you with this notebook file. You also have the definition and trained weights of the model in a file called **HandWrittenDigit-CNN.h5**

**Keras also supports a simpler interface to save both the model weights and model architecture together into a single H5 file, while the HDF5 format store only Model weights and therefore the model architecture is provided as a JSON format.**

- Saving/Loading the model in H5 includes everything we need to know about the model, including:
  - Model weights.
  - Model architecture.

  - Model compilation details (loss and metrics).
  - Model optimizer state.

- This means that we can load and use the model directly, without having to re-compile it.

To upload the HD5 file and use it please procedd as follow :

From the upper toolbar select the *01* icon and Files tab, then drag/drop the file **HandWrittenDigit-CNN.h5** provided in the box folder

Therefore the file appears in the right hand side bar.

Move your cursor on the cell bellow and remove everything (cell fully empty !)

Once done click the drop down arrow of the right hand side window where **HandWrittenDigit-CNN.h5** is and select **insert to code>>Insert Streaming Object**

the equivalent of the following should appear with your own project COS credentials

```
import types
import pandas as pd
from botocore.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.
client_8a2a8e9ef5a44a08aaca7ec89672ecaa = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='xxxxxxxxxxxxx',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

# 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/
streaming_body_1 = client_8a2a8e9ef5a44a08aaca7ec89672ecaa.get_object(Bucket='XXXXXXX', Key='HandWrittenDigit-CNN.h5')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(streaming_body_1, "__iter__"): streaming_body_1.__iter__ = types.MethodType( __iter__, streaming_body_1 )
```

You also need to retrieve the bucket name of you Cloud Object Storage (COS) from the inserted code and then insert it into the cell where you will need to download files from the COS (see sample below)

client*xxxxxxxx.get*object( Bucket='my-generated-bucket-name-123245566788' , Key='HandWrittenDigit-CNN.h5')['Body']

```
client_COS.download_file(Bucket='<inset your bucket-name here>',Key='HandWrittenDigit-CNN.h5',Filename='HandWrittenDigit-CNN.h5')
```

Last but not least rename the variable called **'client_8a2a8e9ef......72ecaa'** with **client_COS** (a bit more clear and reusable for the rest of the notebook !

You're now ready to usethe HD5 definition and weights for your model instead of having to retrain it.

```
import types
import pandas as pd
from botocore.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.
client_8a2a8e9ef5a44a08aaca7ec89672ecaa = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='SgL3gHSfOX7WRMxOLrrvDiDvOl8Z0aCkeMIL9S3j-9Ge',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

# 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/
streaming_body_1 = client_8a2a8e9ef5a44a08aaca7ec89672ecaa.get_object(Bucket='demoai-donotdelete-pr-odc7lk3sakuluh', Key='_mini_XCEPTION.102-0.66.hdf5')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(streaming_body_1, "__iter__"): streaming_body_1.__iter__ = types.MethodType( __iter__, streaming_body_1 )
```

```
Client_COS = client_8a2a8e9ef5a44a08aaca7ec89672ecaa
```

```
ModelFile = 'HandWrittenDigit-CNN.h5'

ReTrainModel = 5
try:
    # Replace the below bucket name by your own bucket project name.
    Client_COS.download_file(Bucket='demoai-donotdelete-pr-odc7lk3sakuluh', Key=ModelFile,Filename=ModelFile)
except:
    # Model never created tbd
    RetrainModel = 1
else:
    RetrainModel = 0
print("Model to be retrain : ", RetrainModel)
!ls
```

```
Model to be retrain :  0
HandWrittenDigit-CNN.h5
```

```
# Define Model

def base_model():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])
    return model
```

```
from keras.models import load_model

if RetrainModel == 0:
    cnn_n = load_model(ModelFile)
    cnn_n.compile(optimizer='adam', loss='categorical_crossentropy')
else:
    cnn_n = base_model()

cnn_n.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776   
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290      
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
```

```
# Vizualizing model structure
sequential_model_to_ascii_printout(cnn_n)
```

```
           OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

               Input   #####     28   28    1
              Conv2D    \|/  -------------------       320     0.0%
                relu   #####     26   26   32
              Conv2D    \|/  -------------------     18496     1.5%
                relu   #####     24   24   64
        MaxPooling2D   Y max -------------------         0     0.0%
                       #####     12   12   64
             Dropout    | || -------------------         0     0.0%
                       #####     12   12   64
             Flatten   ||||| -------------------         0     0.0%
                       #####        9216
               Dense   XXXXX -------------------   1179776    98.3%
                relu   #####         128
             Dropout    | || -------------------         0     0.0%
                       #####         128
               Dense   XXXXX -------------------      1290     0.1%
             softmax   #####          10
```

```
# Fit model
print(y_train.shape)
if RetrainModel == 1:
    cnn = cnn_n.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(x_test, y_test))
```

```
(60000, 10)
```

```
if RetrainModel == 1:
    scores = cnn_n.evaluate(x_test, y_test, verbose=0)
    print(scores)
    print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
if RetrainModel == 1:
    cnn_n.save(ModelFile)
    ClientCOS.upload_file(Bucket='demoai-donotdelete-pr-odc7lk3sakuluh', Key=ModelFile,Filename=ModelFile)
```

## 2.2 Storing the model

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient

wml_client = WatsonMachineLearningAPIClient(WML_CREDENTIALS)
```

```
cnn_n.save("mnist_cnn.h5")
!rm mnist_cnn.tar*
!tar -czvf mnist_cnn.tar.gz mnist_cnn.h5
```

```
rm: cannot remove ‘mnist_cnn.tar*’: No such file or directory
mnist_cnn.h5
```

```
!rm mnist_cnn.h5
```

```
model_name = "MNIST Model"

# Update the FRAMEWORK_VERSION below depending on the tensorflow version used
model_meta = {
    wml_client.repository.ModelMetaNames.NAME: model_name,
    wml_client.repository.ModelMetaNames.DESCRIPTION: "MNIST model",
    wml_client.repository.ModelMetaNames.FRAMEWORK_NAME: "tensorflow",
    wml_client.repository.ModelMetaNames.FRAMEWORK_VERSION: "1.11",
    wml_client.repository.ModelMetaNames.FRAMEWORK_LIBRARIES: [
         {"name": "keras", "version": "2.2.4"}
    ]
}
```

```
 wml_client.repository.list()
# wml_client.repository.delete('')
# wml_client.deployments.delete('')
```

```
------------------------------------  ------------------------------  -----------
-------------  ---------------  -----------------
GUID                                  NAME                            CREATED
               FRAMEWORK        TYPE
13bc8f83-ec48-41e5-ba28-dee6154e2080  Spark German Risk Model - Final  2019-11-08T
10:03:42.944Z  mllib            definition
91e232a5-0bc8-4916-81f7-19be81aa33bf  Spark German Risk Model - Final  2019-10-02T
10:13:10.364Z  mllib            definition
812f030f-56c7-4231-a24e-0d0ed9fb917c  Spark German Risk Model - Final  2019-09-30T
19:39:32.021Z  mllib            definition
85c80a15-a03e-45b6-904d-806a291e260a  Spark German Risk Model - Final  2019-09-30T
17:51:50.181Z  mllib            definition
60d3a487-1f32-4fd4-9a49-7ea85001306d  Spark German Risk Model - Final  2019-09-30T
17:50:29.777Z  mllib            definition
35a9ed96-fc11-4129-9a96-f83f71020694  Spark German Risk Model - Final  2019-09-30T
17:46:37.635Z  mllib            definition
540264f8-7fe8-4ea4-896c-50e00175d9e9  Spark German Risk Model - Final  2019-09-30T
17:45:47.523Z  mllib            definition
83c9b392-20cd-4c63-89c2-882c250a26ff  Spark German Risk Model - Final  2019-09-30T
17:43:26.744Z  mllib            definition
43fb2e21-e4fc-4573-9686-2f674f526610  Spark German Risk Model - Final  2019-09-30T
17:41:27.191Z  mllib            definition
15382117-d170-407d-aea6-9adfc00fbce5  Spark German Risk Model - Final  2019-09-30T
17:28:57.386Z  mllib            definition
71cb4f66-68cf-4305-b17b-c4714d132d2e  Spark German Risk Model - Final  2019-09-29T
11:33:31.607Z  mllib            definition
50d44e6f-282a-45c2-b7f8-be34359b2591  Spark German Risk Model - Final  2019-09-23T
14:10:19.047Z  mllib            definition
0495091a-cacd-43e6-be23-4d80c13c987c  Text Binary Classifier           2019-09-18T
16:13:34.693Z  mllib            definition
```

```
0b87bef5-5e35-42fa-b630-3b04c1dd784a  Text Binary Classifier           2019-09-18T
10:30:17.718Z  mllib            definition
a70ff561-104a-4201-b10d-4d62464086d8  Text Binary Classifier           2019-09-18T
09:09:43.694Z  mllib            definition
e946bfbf-ea9e-46ca-9fe1-240b55ba4743  Text Binary Classifier           2019-09-17T
15:04:37.361Z  mllib            definition
21e378e8-3cd4-4d22-ba53-366bba18e3a2  FER-Model-HDF5                   2019-11-11T
13:55:43.357Z  tensorflow-1.11  model
6c48ad35-8ec4-4b0a-9390-d63d0b8442bf  FER-Model-HDF5                   2019-11-11T
13:34:54.507Z  tensorflow-1.11  model
7a50683b-66dd-4e63-b44f-5c2799dfcdd6  MNIST Model                      2019-11-10T
07:51:53.660Z  tensorflow-1.11  model
5408520e-b01f-46aa-9767-9e690bd02f3e  MNIST Model                      2019-11-08T
14:56:22.452Z  tensorflow-1.11  model
cbe4d658-aba7-4d0b-9bad-831238446281  MNIST Model                      2019-11-08T
11:31:55.317Z  tensorflow-1.11  model
32944fbd-6503-4a1d-9b0b-2cf968a6d169  MNIST Model                      2019-11-08T
11:23:40.638Z  tensorflow-1.11  model
c018330b-44b8-4a7d-9d3c-cbabb7d5e239  MNIST Model                      2019-11-08T
11:13:53.134Z  tensorflow-1.11  model
dbf094de-0d46-48f7-948c-c84f6b8b248a  MNIST Model                      2019-11-08T
11:06:30.001Z  tensorflow-1.11  model
183bcc4d-7169-4ac4-8239-4acc23f592e1  MNIST Model                      2019-11-08T
10:48:37.575Z  tensorflow-1.11  model
dcf1e624-2696-4997-aaa5-35af4fdb4aa4  Spark German Risk Model - Final  2019-11-08T
10:03:50.316Z  mllib-2.3        model
c852301e-c62c-4f7b-ab09-307cb01403f4  MNIST Model                      2019-11-07T
15:52:38.664Z  tensorflow-1.11  model
815f7ce2-632f-4b1a-88d0-7c738d000bf7  MNIST Model                      2019-11-07T
15:20:43.898Z  tensorflow-1.11  model
4f13023a-b135-4cd8-888e-d16bf1fab28c  MNIST Model                      2019-11-07T
15:19:22.070Z  tensorflow-1.11  model
3019b211-19e8-4052-b268-99934c7dacc5  MNIST Model                      2019-11-07T
15:15:10.354Z  tensorflow-1.11  model
dbb150b0-e542-4f15-b5f5-ecb669db1f42  MNIST Model                      2019-11-07T
14:56:41.645Z  tensorflow-1.11  model
fcc006a9-2801-499c-81a3-c89508715e29  MNIST Model                      2019-11-07T
14:44:58.489Z  tensorflow-1.11  model
683bc719-9691-449a-919e-e2e8b2d33f44  MNIST Model                      2019-11-07T
14:35:04.243Z  tensorflow-1.11  model
c035f6ab-11b8-48c7-9a39-c4cd41cac11e  Simpsons300                      2019-11-06T
10:09:44.008Z  tensorflow-1.5   model
8fd859c9-acfa-4bec-a1da-d32726d58cd7  FER-Model-HDF5                   2019-10-16T
20:21:05.302Z  tensorflow-1.11  model
1fb1d948-111e-41ed-af51-c9e34f661f5a  MNIST Model                      2019-10-02T
14:17:54.362Z  tensorflow-1.11  model
ee1bed7b-6eb4-44e9-a5a0-f4ff995de644  FER-Kaggle                       2019-09-26T
12:40:59.874Z  tensorflow-1.5   model
```

```
c62f61ca-67dd-41d5-8c06-faff03f6c883  MNIST Model                       2019-09-25T
12:52:56.362Z  tensorflow-1.11  model
390d32b2-b3f0-4f67-aceb-6d9b04b0db0a  Text Binary Classifier            2019-09-18T
16:13:50.846Z  mllib-2.3        model
35ff37c5-b12b-437b-b269-809cdd815e27  Text Binary Classifier            2019-09-18T
09:09:49.035Z  mllib-2.3        model
3f0f33c8-42a8-46a2-98f9-2407bbca2511  Text Binary Classifier            2019-09-17T
15:04:42.888Z  mllib-2.3        model
4b540dfa-400d-4160-a399-f72d166409a5  GermanCreditRiskModel             2019-09-16T
12:51:34.237Z  mllib-2.3        model
f4f0a9ff-1a19-4d95-bb63-df884413f52f  FER-Model-HDF5 Deployment         2019-11-11T
13:55:45.679Z  tensorflow-1.11  online deployment
448084c5-1e3d-4485-9c91-df88a2b6ab78  FER-Model-HDF5 Deployment         2019-11-11T
13:34:56.918Z  tensorflow-1.11  online deployment
8947f73d-5f5e-4353-a1cb-7a18d24223f0  MNIST Model Deployment            2019-11-10T
07:51:56.304Z  tensorflow-1.11  online deployment
53f2dddb-be2b-4ca6-a5fc-0b60a4ccd770  MNIST Model Deployment            2019-11-08T
14:56:25.368Z  tensorflow-1.11  online deployment
6541e80d-f70b-4991-8af0-46fb1314855f  MNIST Model Deployment            2019-11-08T
11:31:58.204Z  tensorflow-1.11  online deployment
91b8f685-7c35-4c49-b0f4-1970838a53c4  MNIST Model Deployment            2019-11-08T
11:23:43.449Z  tensorflow-1.11  online deployment
52f3bd15-41db-4b35-a82f-c2176803a711  MNIST Model Deployment            2019-11-08T
11:13:55.702Z  tensorflow-1.11  online deployment
df59a0b8-f09c-4422-9884-e83dca7c6006  MNIST Model Deployment            2019-11-08T
11:06:32.491Z  tensorflow-1.11  online deployment
------------------------------------  -------------------------------  -----------
-------------  ---------------  -----------------
Note: Only first 50 records were displayed. To display more use more specific list
 functions.
```

```
# published_model_details = wml_client.repository.get_details('7d7d20b6-6e54-4643-
ae14-f99f41f0f986')
```

```
published_model_details = wml_client.repository.store_model(model='mnist_cnn.tar.g
z', meta_props=model_meta)
```

```
Note: Model of framework tensorflow and versions 1.5/1.11 has been deprecated. The
se versions will not be supported after 26th Nov 2019.
```

```
model_uid = wml_client.repository.get_model_uid(published_model_details)
model_uid
```

```
'0350bda8-6d1a-4763-a0a4-9c7070527ad7'
```

## 2.3 Deploying the model

```
deployment= wml_client.deployments.create(name= model_name + " Deployment", model_
uid=model_uid)
```

```
Note: Model of framework tensorflow and versions 1.5/1.11 has been deprecated. The
se versions will not be supported after 26th Nov 2019.



##################################################################################
#####

Synchronous deployment creation for uid: '0350bda8-6d1a-4763-a0a4-9c7070527ad7' st
arted

##################################################################################
#####


INITIALIZING
DEPLOY_IN_PROGRESS..
DEPLOY_SUCCESS



----------------------------------------------------------------------------------
--------------
Successfully finished deployment creation, deployment_uid='eb5b0436-a33a-4297-92db
-8a2d3126ee86'
----------------------------------------------------------------------------------
--------------
```

```
scoring_url = wml_client.deployments.get_scoring_url(deployment)
print(scoring_url)
```

```
https://us-south.ml.cloud.ibm.com/v3/wml_instances/febb80c2-33af-4014-8dd8-ef2170f
f4cfb/deployments/eb5b0436-a33a-4297-92db-8a2d3126ee86/online
```

# 3. Subscriptions

## 3.1 Configuring OpenScale

```
from ibm_ai_openscale import APIClient
from ibm_ai_openscale.engines import WatsonMachineLearningAsset

aios_client = APIClient(AIOS_CREDENTIALS)
aios_client.version
```

```
'2.1.17'
```

```
#  CLEAN SUBSCRIPTION ENTRIES
subscriptions_uids = aios_client.data_mart.subscriptions.get_uids()
for subscription in subscriptions_uids:
    sub_name = aios_client.data_mart.subscriptions.get_details(subscription)['enti
ty']['asset']['name']
    if sub_name == model_name:
        aios_client.data_mart.subscriptions.delete(subscription)
        print('Deleted existing subscription for', model_name)
```

```
Deleted existing subscription for MNIST Model
```

## 3.2 Subscribe the asset

```
from ibm_ai_openscale.supporting_classes import *

aios_client.data_mart.subscriptions.list()
# aios_client.data_mart.subscriptions.delete('657c48a9-d29a-4e29-a215-b8a28046bfd3
')

Asset = WatsonMachineLearningAsset(model_uid,
                                   problem_type=ProblemType.MULTICLASS_CLASSIFICAT
ION,
                                   input_data_type=InputDataType.UNSTRUCTURED_IMAG
E,
                                   probability_column='probability'
                                  )
subscription = aios_client.data_mart.subscriptions.add(Asset)
```

## Subscriptions

| uid | name | type | binding_uid | created |
|---|---|---|---|---|
| ba8e4e44-5b90-459d-9aa8-fe04631e15e4 | FER-Model-HDF5 | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-11-11T15:26:14.501Z |
| 087a04a9-2318-472e-ad42-3783b631666b | Spark German Risk Model - Final | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-11-08T10:04:43.254Z |
| b5079da2-264b-43e8-a71f-a9ee23208832 | FER-Kaggle | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-10-16T11:26:28.635Z |
| c50ada6b-a76e-42be-b000-831d519dda63 | FER-2013 | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-10-09T22:18:02.029Z |

```
aios_client.data_mart.subscriptions.list()
```

## Subscriptions

| uid | name | type | binding_uid | created |
|---|---|---|---|---|
| e56ffa07-970d-4d74-b284-1e1e03244544 | MNIST Model | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-11-12T14:14:59.425Z |
| ba8e4e44-5b90-459d-9aa8-fe04631e15e4 | FER-Model-HDF5 | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-11-11T15:26:14.501Z |
| 087a04a9-2318-472e-ad42-3783b631666b | Spark German Risk Model - Final | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-11-08T10:04:43.254Z |
| b5079da2-264b-43e8-a71f-a9ee23208832 | FER-Kaggle | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-10-16T11:26:28.635Z |
| c50ada6b-a76e-42be-b000-831d519dda63 | FER-2013 | model | febb80c2-33af-4014-8dd8-ef2170ff4cfb | 2019-10-09T22:18:02.029Z |

```
subscription.get_details()
```

```
{'entity': {'asset': {'asset_id': '0350bda8-6d1a-4763-a0a4-9c7070527ad7',
   'asset_type': 'model',
   'created_at': '2019-11-12T14:14:22.580Z',
   'name': 'MNIST Model',
   'url': 'https://us-south.ml.cloud.ibm.com/v3/wml_instances/febb80c2-33af-4014-8
dd8-ef2170ff4cfb/published_models/0350bda8-6d1a-4763-a0a4-9c7070527ad7'},
  'asset_properties': {'input_data_type': 'unstructured_image',
   'model_type': 'tensorflow-1.11',
   'probability_fields': ['probability'],
   'problem_type': 'multiclass',
   'runtime_environment': 'None Provided'},
  'configurations': [{'enabled': True,
    'monitor_definition_id': 'payload_logging',
    'type': 'payload_logging',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/payload_logging'},
   {'enabled': False,
    'monitor_definition_id': 'explainability',
    'type': 'explainability',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/explainability'},
   {'enabled': True,
    'monitor_definition_id': 'performance_monitoring',
    'type': 'performance_monitoring',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/performance_monitoring'},
   {'enabled': False,
    'monitor_definition_id': 'fairness_monitoring',
    'type': 'fairness_monitoring',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/fairness_monitoring'},
   {'enabled': False,
    'monitor_definition_id': 'correlations',
    'type': 'correlations',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/correlations'},
   {'enabled': False,
    'monitor_definition_id': 'drift',
    'type': 'drift',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/drift'},
   {'enabled': False,
    'monitor_definition_id': 'quality',
    'type': 'quality_monitoring',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/quality'},
   {'enabled': False,
    'monitor_definition_id': 'my_model_performance',
    'type': 'my_model_performance',
    'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/f
ebb80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e0324
4544/configurations/my_model_performance'}],
  'deployments': [{'created_at': '2019-11-12T14:14:22.640Z',
    'deployment_id': 'eb5b0436-a33a-4297-92db-8a2d3126ee86',
    'deployment_rn': '',
    'deployment_type': 'online',
    'name': 'MNIST Model Deployment',
    'scoring_endpoint': {'request_headers': {'Content-Type': 'application/json'},
     'url': 'https://us-south.ml.cloud.ibm.com/v3/wml_instances/febb80c2-33af-4014
-8dd8-ef2170ff4cfb/deployments/eb5b0436-a33a-4297-92db-8a2d3126ee86/online'},
    'url': 'https://us-south.ml.cloud.ibm.com/v3/wml_instances/febb80c2-33af-4014-
8dd8-ef2170ff4cfb/deployments/eb5b0436-a33a-4297-92db-8a2d3126ee86'}],
  'service_binding_id': 'febb80c2-33af-4014-8dd8-ef2170ff4cfb',
  'status': {'state': 'active'}},
 'metadata': {'guid': 'e56ffa07-970d-4d74-b284-1e1e03244544',
  'url': '/v1/data_marts/70ee9046-f34e-441c-8dbe-75d57d88b6f7/service_bindings/feb
b80c2-33af-4014-8dd8-ef2170ff4cfb/subscriptions/e56ffa07-970d-4d74-b284-1e1e032445
44',
  'created_at': '2019-11-12T14:14:59.425Z'}}
```

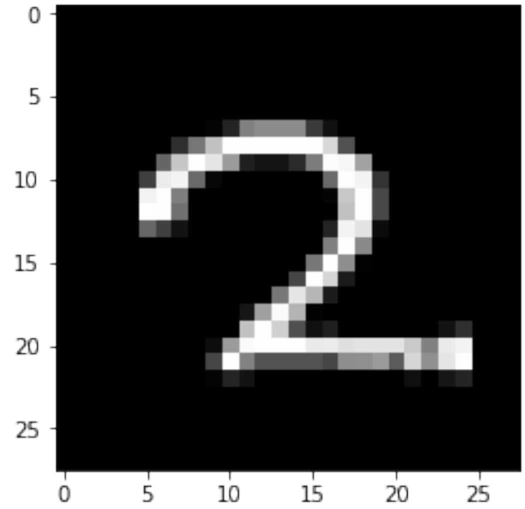
## 3.3 Score the model and get transaction-id

```
!pip install numpy
!pip install matplotlib

import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
img = np.array(x_test[77], dtype='float')
pixels = img.reshape((28, 28))
plt.imshow(pixels, cmap='gray')
plt.show()
```

```
Requirement already satisfied: numpy in /opt/conda/envs/Python36/lib/python3.6/site-packages (1.15.4)
Requirement already satisfied: matplotlib in /opt/conda/envs/Python36/lib/python3.6/site-packages (3.0.2)
Requirement already satisfied: numpy>=1.10.0 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from matplotlib) (1.15.4)
Requirement already satisfied: cycler>=0.10 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from matplotlib) (1.0.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from matplotlib) (2.3.1)
Requirement already satisfied: python-dateutil>=2.1 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from matplotlib) (2.7.5)
Requirement already satisfied: six in /opt/conda/envs/Python36/lib/python3.6/site-packages (from cycler>=0.10->matplotlib) (1.12.0)
Requirement already satisfied: setuptools in /opt/conda/envs/Python36/lib/python3.6/site-packages (from kiwisolver>=1.0.1->matplotlib) (39.1.0)
```



```
import json

Wait(20)

scoring_data = {'values': [x_test[77].tolist()]}
predictions = wml_client.deployments.score(scoring_url, scoring_data)
print(json.dumps(predictions, sort_keys=True, indent=4))
```

```
 {
    "fields": [
        "prediction",
        "prediction_classes",
        "probability"
    ],
    "values": [
        [
            [
                3.840007047983818e-05,
                3.6366909625940025e-06,
                0.9999328851699829,
                1.4511347501411365e-07,
                2.573256274729374e-09,
                3.4113309399508296e-10,
                3.763992673100347e-09,
                2.2245205400395207e-05,
                2.4453431706206175e-06,
                2.6317934498365503e-07
            ],
            2,
            [
                3.840007047983818e-05,
                3.6366909625940025e-06,
                0.9999328851699829,
                1.4511347501411365e-07,
                2.573256274729374e-09,
                3.4113309399508296e-10,
                3.763992673100347e-09,
                2.2245205400395207e-05,
                2.4453431706206175e-06,
                2.6317934498365503e-07
            ]
        ]
    ]
}
```

```
Wait(20)
transaction_id = subscription.payload_logging.get_table_content().scoring_id[0]
transaction_id
```

```
'648e088bab6e54d81303cc1744a03233-1'
```

# 4. Explainability

## 4.1 Configure Explainability

```
subscription.explainability.enable()
```

```
subscription.explainability.get_details()
```

```
{'enabled': True}
```

## 4.2 Get explanation for the transaction

```
explanation = ()
try :
    explanation = subscription.explainability.run(transaction_id, background_mode=
False,cem=False)
except:
  print("Something went wrong")
  wml_client.repository.delete(model_uid)
  deployment_id = wml_client.deployments.get_uid(deployment)
  wml_client.deployments.delete(deployment_id)
```

```
=================================================================

 Looking for explanation for 648e088bab6e54d81303cc1744a03233-1

=================================================================




in_progress.........
finishedSomething went wrong
{"trace":"c0de5ec42ececc4981bd7cd3a9c13a76","errors":[{"code":"not_found","message
":"Requested object could not be found."}]}
```

If you get an error in the preious cell something ending by **KeyError: 'cem_state'** it's a bug :( in the library, but still you can collect the transaction_id from the upper cell

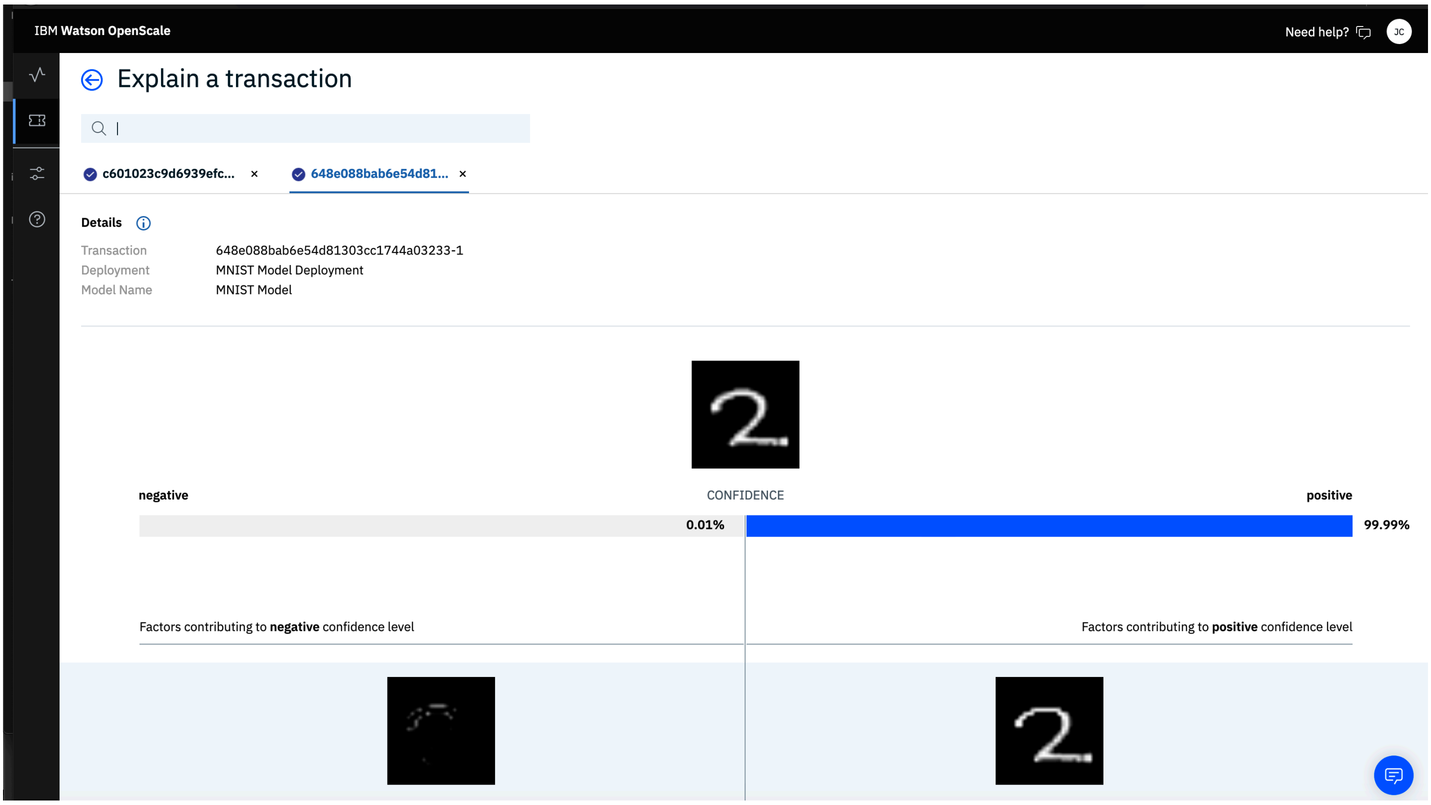
```
# Wait(60)
transaction_id = subscription.payload_logging.get_table_content().scoring_id[0]
transaction_id
```

then open the following webpage and paste the transaction_id and search for it, you will see the result of the image model explainability.

https://aiopenscale.cloud.ibm.com/aiopenscale/explain

### Explaining image model transactions

For an image classification model example of explainability, you can see which parts of an image contributed positively to the predicted outcome and which contributed negatively. In the following example, the image in the positive pane shows the parts which impacted positively to the prediction and the image in the negative pane shows the parts of images that had a negative impact on the outcome.



Explain a transaction

Details

Transaction 648e088bab6e54d81303cc1744a03233-1
Deployment MNIST Model Deployment
Model Name MNIST Model

Factors contributing to **negative** confidence level

Factors contributing to **positive** confidence level

```
import json
print (json.dumps(explanation,  sort_keys=True, indent=4))
```

## The explanation images can be obtained using the cells below

```
!pip install Pillow
from PIL import Image
import base64
import io

imgOrigin = explanation["entity"]["predictions"][0]["explanation_features"][0]["fu
ll_image"]
img_data = base64.b64decode(imgOrigin)
OriginPic = Image.open(io.BytesIO(img_data)).resize((128, 128)).convert('RGBA')
```

```
img = explanation["entity"]["predictions"][1]["explanation_features"][0]["full_image"]
img_data = base64.b64decode(img)
ExpPic = Image.open(io.BytesIO(img_data)).resize((128, 128))
```

```
Background = ExpPic.convert('RGBA')

# "data" is a height x width x 4 numpy array
data = np.array(Background)

# Temporarily unpack the bands for readability
red, green, blue, alpha = data.T

# Replace white with red... (leaves alpha values alone...)
white_areas = (red != 0) | (blue != 0) | (green != 0)
data[..., :-1][white_areas.T] = (255, 0, 0) # Transpose back needed

Background = Image.fromarray(data)
```

```
Image.blend(Background, OriginPic,alpha=0.3).resize((256,256))
```

```
wml_client.repository.delete(model_uid)
deployment_id = wml_client.deployments.get_uid(deployment)
wml_client.deployments.delete(deployment_id)
```