## Import necessary libraries

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

import keras
from keras.models import Sequential
from keras.layers import Conv2D, Lambda, MaxPooling2D
from keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import BatchNormalization

from keras.preprocessing.image import ImageDataGenerator

from keras.utils.np_utils import to_categorical

from keras.datasets import mnist
```

## Load the data

```
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
print(X_train.shape)
print(X_test.shape)
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
(60000, 28, 28)
(10000, 28, 28)
```

## Data pre-processing

```
X_train = X_train / 255.0
X_test = X_test / 255.0
X_train = X_train.reshape(-1,28,28,1)
X_test = X_test.reshape(-1,28,28,1)
Y_train = to_categorical(Y_train)
Y_test = to_categorical(Y_test)
mean = np.mean(X_train)
std = np.std(X_train)

def standardize(x):
    return (x-mean)/std
```

## Create model

```
model=Sequential()
```

```
model.add(Conv2D(filters=64, kernel_size = (3,3), activation="relu",
input_shape=(28,28,1)))
model.add(Conv2D(filters=64, kernel_size = (3,3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())

model.add(Conv2D(filters=128, kernel_size = (3,3), activation="relu"))
model.add(Conv2D(filters=128, kernel_size = (3,3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())

model.add(Conv2D(filters=256, kernel_size = (3,3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(512,activation="relu"))

model.add(Dense(10,activation="softmax"))

model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 12, 12, 64)       0
 )

 batch_normalization (BatchN  (None, 12, 12, 64)       256
 ormalization)

 conv2d_2 (Conv2D)           (None, 10, 10, 128)       73856

 conv2d_3 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 batch_normalization_1 (Batc  (None, 4, 4, 128)        512
 hNormalization)

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         295168

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 batch_normalization_2 (Batc  (None, 1, 1, 256)        1024
 hNormalization)

 flatten (Flatten)           (None, 256)               0
```

```
 dense (Dense)               (None, 512)               131584

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 692,682
Trainable params: 691,786
Non-trainable params: 896
_________________________________________________________________
```

## Define training parameters

```
datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=15,
        zoom_range = 0.01,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=False,
        vertical_flip=False)


train_gen = datagen.flow(X_train, Y_train, batch_size=128)
test_gen = datagen.flow(X_test, Y_test, batch_size=128)
epochs = 10
batch_size = 128
train_steps = X_train.shape[0] // batch_size
valid_steps = X_test.shape[0] // batch_size

es = keras.callbacks.EarlyStopping(
        monitor="val_accuracy",
        patience=10,
        verbose=1,
        mode="max",
        restore_best_weights=True,
    )

rp = keras.callbacks.ReduceLROnPlateau(
        monitor="val_accuracy",
        factor=0.2,
        patience=3,
        verbose=1,
        mode="max",
        min_lr=0.00001,
    )
```

## Train the model

```
history = model.fit(train_gen,
                    epochs = epochs,
                    steps_per_epoch = train_steps,
```

```
                    validation_data = test_gen,
                    validation_steps = valid_steps,
                    callbacks=[es, rp])
Epoch 1/10
468/468 [==============================] - 428s 912ms/step - loss: 0.1275 -
accuracy: 0.9597 - val_loss: 0.1187 - val_accuracy: 0.9662 - lr: 0.0010
Epoch 2/10
468/468 [==============================] - 430s 920ms/step - loss: 0.0548 -
accuracy: 0.9837 - val_loss: 0.0578 - val_accuracy: 0.9832 - lr: 0.0010
Epoch 3/10
468/468 [==============================] - 424s 906ms/step - loss: 0.0406 -
accuracy: 0.9873 - val_loss: 0.0435 - val_accuracy: 0.9877 - lr: 0.0010
Epoch 4/10
468/468 [==============================] - 406s 868ms/step - loss: 0.0369 -
accuracy: 0.9888 - val_loss: 0.0311 - val_accuracy: 0.9903 - lr: 0.0010
Epoch 5/10
468/468 [==============================] - 411s 878ms/step - loss: 0.0341 -
accuracy: 0.9897 - val_loss: 0.0277 - val_accuracy: 0.9914 - lr: 0.0010
Epoch 6/10
468/468 [==============================] - 412s 879ms/step - loss: 0.0309 -
accuracy: 0.9910 - val_loss: 0.0317 - val_accuracy: 0.9900 - lr: 0.0010
Epoch 7/10
468/468 [==============================] - 411s 878ms/step - loss: 0.0309 -
accuracy: 0.9910 - val_loss: 0.0317 - val_accuracy: 0.9900 - lr: 0.0010
Epoch 8/10
468/468 [==============================] - ETA: 0s - loss: 0.0271 - accuracy:
0.9914
Epoch 00008: ReduceLROnPlateau reducing learning rate to
0.00020000000949949026.
468/468 [==============================] - 408s 873ms/step - loss: 0.0271 -
accuracy: 0.9914 - val_loss: 0.0336 - val_accuracy: 0.9891 - lr: 0.0010
Epoch 9/10
468/468 [==============================] - 410s 875ms/step - loss: 0.0155 -
accuracy: 0.9952 - val_loss: 0.0175 - val_accuracy: 0.9948 - lr: 2.0000e-04
Epoch 10/10
468/468 [==============================] - 412s 881ms/step - loss: 0.0118 -
accuracy: 0.9961 - val_loss: 0.0156 - val_accuracy: 0.9945 - lr: 2.0000e-04
```

## Save the model

```
model.save("model.h5")
!tar -zcvf model.tgz model.h5
model.h5
```

## Install necessary packages

```
!pip install watson-machine-learning-client
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538
kB)
     |████████████████████████████████| 538 kB 15.0 MB/s eta 0:00:01
Requirement already satisfied: certifi in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(2022.9.24)
```

```
Requirement already satisfied: lomond in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
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)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(1.26.7)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(1.18.21)
Requirement already satisfied: pandas in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(4.62.3)
Requirement already satisfied: requests in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from watson-machine-learning-client)
(2.26.0)
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)
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)
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)
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)
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)
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)
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)
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)
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)
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)
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)
Installing collected packages: watson-machine-learning-client
Successfully installed watson-machine-learning-client-1.0.391
```

## Connect to IBM Watson Machine Learning instance

```
from ibm_watson_machine_learning import APIClient

API_KEY = ""

credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": API_KEY
}

client = APIClient(credentials)
def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']['name']
== space_name)['metadata']['id'])
space_uid = guid_from_space_name(client, 'Handwritten Digit Recognition')
print("Space UID: ", space_uid)
Space UID:  de0e9dab-efb5-4473-8fc8-3c6e43d91804
client.set.default_space(space_uid)
'SUCCESS'
```

## Define model specifications for deployment

```
client.software_specifications.list()
------------------------------  ------------------------------------  ----
NAME                            ASSET_ID                              TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6           0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                      0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base
runtime-22.1-py3.9              12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                    1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6          1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6            1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt   1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6            1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
spark-mllib_3.2                 20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
tensorflow_2.4-py3.8-horovod    217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda         26215f05-08c3-5a41-a1b0-da66306ce658  base
do_py3.8                        295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8             2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6           2b73a275-7cbf-420b-a912-eae7f436e0bc  base
kernel-spark3.3-py3.9           2b7961e2-e3b1-5a8c-a491-482c8368839a  base
pytorch_1.2-py3.6               2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3                 2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt      32983cea-3f32-4400-8965-dde874a8d67e  base
```

```
spark-mllib_3.0-py37           36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4                390d21f8-e58b-4fac-9c55-d7ceda621326  base
xgboost_0.82-py3.6             39e31acd-5f30-41dc-ae44-60233c80306e  base
pytorch-onnx_1.2-py3.6-edt     40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
default_r36py38                41c247d3-45f8-5a71-b065-8580229facf0  base
autoai-ts_rt22.1-py3.9         4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                 42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                   493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6          49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6             4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6         50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8            52c57136-80fa-572e-8728-a5e7cbb42cde  base
spark-mllib_2.4-scala_2.11     55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0                5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                 5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1              5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                     5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
autoai-kb_3.1-py3.7            632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8         634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
spark-mllib_2.3-r_3.6          6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c  base
tensorflow_2.4-py3.7           65e171d7-72d1-55d9-8ebb-f813d620c9bb  base
spss-modeler_18.2              687eddc9-028a-4117-b9dd-e57b36f1efa5  base
-----------------------------  ------------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit'
parameter.
software_spec_uid = client.software_specifications.get_uid_by_name("runtime-
22.1-py3.9")
software_spec_uid
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
model_details = client.repository.store_model(model="model.tgz", meta_props={
    client.repository.ModelMetaNames.NAME: "CNN",
    client.repository.ModelMetaNames.TYPE: "tensorflow_2.7",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
})

model_id = client.repository.get_model_id(model_details)
model_id
'c7fd0556-1d58-4698-b9bb-23106e4e6bc5'
```

## Download the deployed model

```
client.repository.download(model_id, "model.tar.gz")
Successfully saved model content to file: 'model.tar.gz'
'/home/wsuser/work/model.tar.gz'
```