

A Novel Method for Handwritten Digit Recognition

Team ID: PNT2022TMID35466

IBM CLOUD HOME PAGE

The screenshot shows the IBM Watson Studio home page. At the top, there's a navigation bar with the IBM Watson Studio logo, a search bar, and user account information (Gayathiri N's Account, Dallas, and a profile icon). Below the navigation bar, a large banner area welcomes the user and provides three main action paths: 'Take a tutorial', 'Work with data', and 'Learn what's new'. To the right of these paths is a 3D visualization of data cubes and a line chart. Below the banner, there are four main sections: 'Quick start' with links to create data pipelines, build customer profiles, catalog and govern data, and build and manage ML models; 'Projects' showing a project titled 'A Novel Method for Handwritten Digit Recognition System' created today at 03:26 PM; 'Notifications' showing an 'Online deployment ready' notification for the same project; and 'Deployments' showing a deployment of the 'Handwritten Digit Recognition System' created today at 05:40 PM.

PROJECT DETAILS

The screenshot shows the 'Projects' page in IBM Watson Studio. At the top, there's a navigation bar with the IBM Watson Studio logo, a search bar, and user account information (Gayathiri N's Account, Dallas, and a profile icon). Below the navigation bar, the page title 'Projects' is displayed. A search bar with the placeholder text 'Find a project' is located below the title. To the right of the search bar is a 'New project' button. Below the search bar, there's a table with the following columns: 'Name', 'Date created', 'Your role', and 'Collaborators'. The table contains one project entry: 'A Novel Method for Handwritten Digit Recognition System', created '4 hours ago', with the role 'Admin' and collaborator 'GN'. A 'New project' button is also located at the bottom right of the table.

Name	Date created	Your role	Collaborators
A Novel Method for Handwritten Digit Recognition System	4 hours ago	Admin	GN

PROJECT DESCRIPTION

Projects / A Novel Method for Handwritten ...

OverviewAssetsJobsManage

Project

General

Access control

Environments

Resource usage

Services & integrations

General

Details

Name

A Novel Method for Handwritten Digit Recognition System

Description

A model to make computer to identify and understand handwritten digits or characters automatically. We use neural networks to train these images and build a deep learning model. Web application is created where the user can upload an image of a handwritten digit. this image is analyzed by the model and the detected result is returned on to user interface.

Project ID

68997eaa-e220-40da-9b84-c6d06ae010bb

Storage

Storage used

16.24 MB

Bucket

anovelmethoforhandwrittendigitre-donotdelete-pr-plg8tc2byxoodd

Manage in IBM Cloud

Danger zone

Leave project

Remove your ability to access this project and its assets.

Leave

PROJECT ASSETS

Projects / A Novel Method for Handwritten ...

OverviewAssetsJobsManage

Find assets

Import assets

New asset

3 assets

All assets

Asset types

Data

Notebooks

All assets

Name	Last modified
Handwritten Digit Recognition Notebook	1 hour ago Modified by you
data.zip application/x-zip-compressed	2 hours ago Modified by you
IBM_Deployment Notebook	2 hours ago Modified by you

IMPORTING REQUIRED LIBRARIES:

Importing Required Libraries

```
In [1]: import numpy as np
import pandas as pd
import random
import tensorflow as tf
import keras
from keras import layers
keras.backend.set_image_data_format('channels_last')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

LOADING INPUT DATASET

```
In [2]: import os, 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.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='l1SLNKuyTOTvyxuymCK02Yj3GQDr1AxjSLgrZRYD5R2G',
    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 = 'handwrittendigitrecognition-donotdelete-pr-yvxxug6r0c2f8y'
object_key = 'data.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 [3]: from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_1.read()), 'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

CNN MODEL:

```
: def CNN():
    model = keras.Sequential()
    # CONV > CONV > BN > RELU > MAXPOOLING > DROPOUT
    model.add(layers.Conv2D(32, (3, 3), (1, 1), padding='valid', input_shape=(28, 28, 1), name='conv2d_1_1'))
    model.add(layers.Conv2D(32, (3, 3), (1, 1), padding='same', name='conv2d_1_2'))
    model.add(layers.BatchNormalization(name='bn_1'))
    model.add(layers.Activation('relu', name='relu_1'))
    model.add(layers.MaxPooling2D((2, 2), (2, 2), padding='valid', name='mp2d_1'))
    model.add(layers.Dropout(0.2, name='drop_1'))
    # CONV > CONV > BN > RELU > MAXPOOLING > DROPOUT
    model.add(layers.Conv2D(64, (3, 3), (1, 1), padding='valid', name='conv2d_2_1'))
    model.add(layers.Conv2D(64, (3, 3), (1, 1), padding='same', name='conv2d_2_2'))
    model.add(layers.BatchNormalization(name='bn_2'))
    model.add(layers.Activation('relu', name='relu_2'))
    model.add(layers.MaxPooling2D((2, 2), (2, 2), padding='valid', name='mp2d_2'))
    model.add(layers.Dropout(0.2, name='drop_2'))
    # FLATTEN > DENSE > CLASSIFICATION
    model.add(layers.Flatten())
    model.add(layers.Dense(100, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))

    return model

: model = CNN()

: model.compile(optimizer = "Adam", loss = "CategoricalCrossentropy", metrics = "accuracy")

: model.summary()
```

TRAINING THE MODEL

```
n [36]: training = model.fit(X_train, Y_train, validation_data = (X_val, Y_val), batch_size = 64, epochs = 5, verbose = 1 )

Epoch 1/5
525/525 [=====] - 90s 170ms/step - loss: 0.2151 - accuracy: 0.9339 - val_loss: 0.0720 - val_accuracy: 0.9771
Epoch 2/5
525/525 [=====] - 89s 170ms/step - loss: 0.0715 - accuracy: 0.9778 - val_loss: 0.0762 - val_accuracy: 0.9767
Epoch 3/5
525/525 [=====] - 88s 167ms/step - loss: 0.0538 - accuracy: 0.9829 - val_loss: 0.0660 - val_accuracy: 0.9805
Epoch 4/5
525/525 [=====] - 89s 169ms/step - loss: 0.0458 - accuracy: 0.9850 - val_loss: 0.0416 - val_accuracy: 0.9889
Epoch 5/5
525/525 [=====] - 89s 169ms/step - loss: 0.0388 - accuracy: 0.9878 - val_loss: 0.0578 - val_accuracy: 0.9832
```

IBM Watson Studio

Search in your workspaces

Buy

?

1

Gayathiri N's Account

Dallas

GN

Projects / A Novel Method for Handwritten ... / Handwritten Digit Recognition

In [45]:

from ibm_watson_machine_learning import APIClient
wml_credentials={
 "url":"https://us-south.ml.cloud.ibm.com",
 "apikey":"PyE-y2TaHIBuZgIBqLmXzt9ob6mYwOt_e3TZWjABvEuQ"
}
client=APIClient(wml_credentials)

In [46]:

def guid_from_space_name(client,space_name):
 spaces=client.spaces.get_details()
 return (next(item for item in space['resources'] if item['entity']['name']==space_name)['metadata']['id'])

In [47]:

space_uid=guid_from_space_name(client,'Handwritten Digit Recognition System')
print("Space UID = "+space_uid)

Space UID = 63a199a8-b8f5-41c3-b00b-db4b853a8752

In [48]:

client.set.default_space(space_uid)

Out[48]:

'SUCCESS'

TESTING THE MODEL:

IBM Watson Studio

Search in your workspaces

Buy

?

1

Gayathiri N's Account

Dallas

GN

Projects / A Novel Method for Handwritten ... / Handwritten Digit Recognition

In [59]:

def predict(model, X, start,end):
 s = int(np.sqrt(end-start))
 fig, ax = plt.subplots(s, s, sharex=True, sharey=True, figsize=(15, 15))
 ax = ax.flatten()
 preds = model.predict(X[start:end])
 for i in range(end-start):
 y_pred = np.argmax(preds[i])
 img = X[start+i].reshape(28, 28)
 ax[i].imshow(img, cmap='Greys', interpolation='nearest')
 ax[i].set_title(f'p: {y_pred}')

In [61]: predict(model, X_test, 75,100)