A Novel Method for Handwritten Digit Recognition
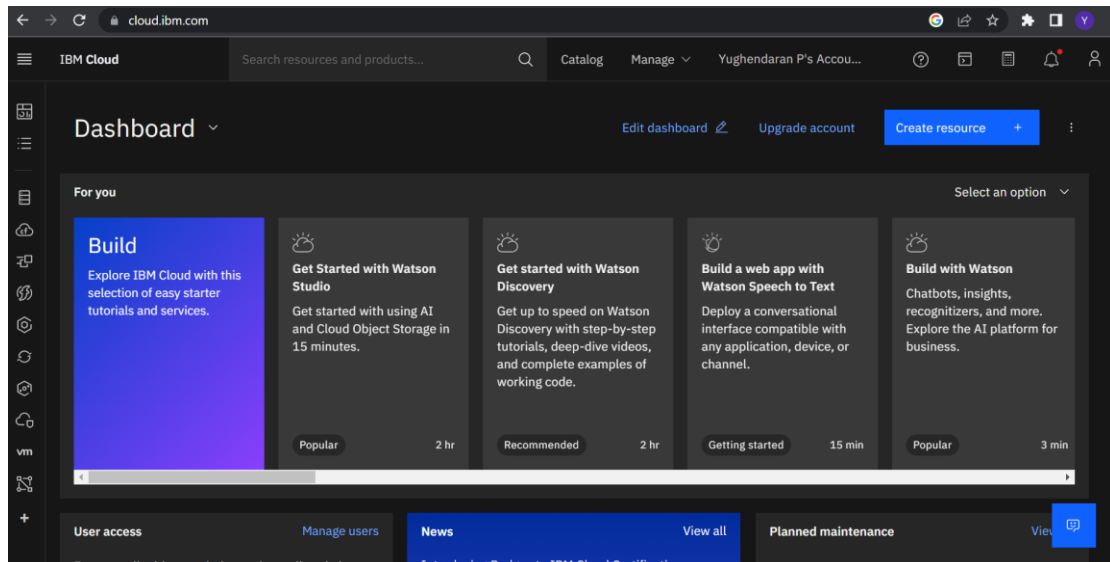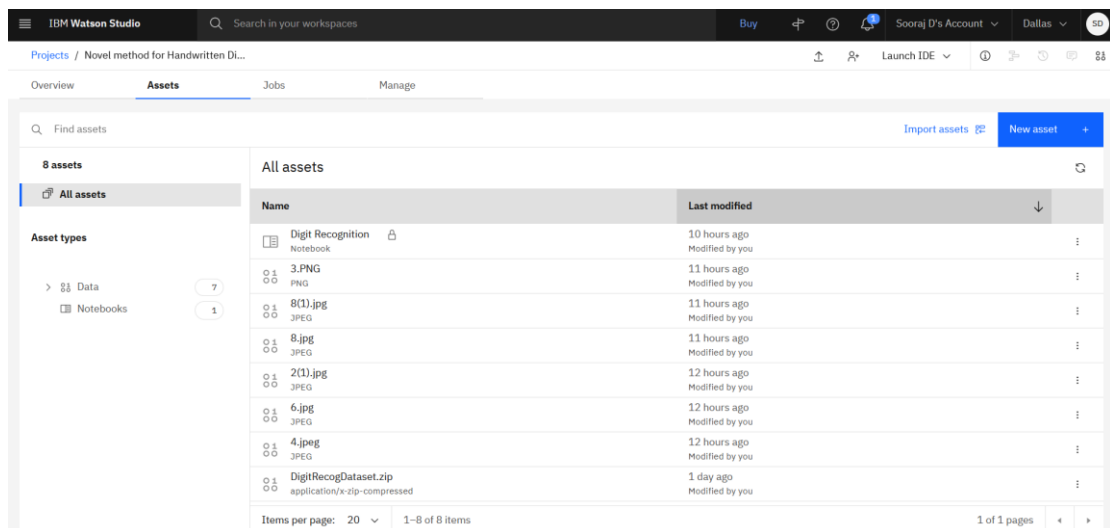
Team ID: PNT2022TMID35418

IBM Cloud Homepage:

IBM Watson Studio:

IBM Notebook:

LOADING THE DATASET

```python
In [66]: 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='g8zhqVmSpbZtmnQORKLSJZn4MX6TmWG1emMZlZSCOKpB',
             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 = 'novelmethodforhandwrittendigitrec-donotdelete-pr-sb2ogtldrhegen'
         object_key = 'DigitRecogDataset.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/
```

```python
In [67]: 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)
```

```python
In [68]: train = pd.read_csv("/home/wsuser/work/DigitRecogDataset/train.csv")
```
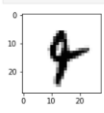
DATA PREPROCESSING

```python
In [72]: X_train = X_train/255.0
         X_test = X_test/255.0

         X_train = X_train.values.reshape(-1, 28, 28, 1)
         X_test = X_test.values.reshape(-1, 28, 28, 1)

         fig, ax = plt.subplots(figsize=(2,2))
         plt.imshow(X_train[random.randint(0,len(X_train)), :, :, 0], cmap='Greys', interpolation='nearest')
         plt.show()
```


TRAIN AND TEST SPLIT

```python
In [73]: validation_size = 0.2
         X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = validation_size)
```

ONE HOT ENCODING

```python
In [74]: Y_train = keras.utils.np_utils.to_categorical(Y_train, num_classes = 10)
         Y_val = keras.utils.np_utils.to_categorical(Y_val, num_classes = 10)
```

MODEL BUILDING

TRAINING THE MODEL

```python
In [125.. model.fit(X_train, Y_train, validation_data = (X_val, Y_val), epochs = 50)

         Epoch 1/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0086 - accuracy: 0.9974 - val_loss: 0.0939 - val_accuracy: 0.9882
         Epoch 2/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0069 - accuracy: 0.9978 - val_loss: 0.0968 - val_accuracy: 0.9875
         Epoch 3/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0056 - accuracy: 0.9985 - val_loss: 0.0934 - val_accuracy: 0.9874
         Epoch 4/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0068 - accuracy: 0.9979 - val_loss: 0.0835 - val_accuracy: 0.9869
         Epoch 5/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0055 - accuracy: 0.9982 - val_loss: 0.0982 - val_accuracy: 0.9885
         Epoch 6/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0071 - accuracy: 0.9979 - val_loss: 0.1012 - val_accuracy: 0.9873
         Epoch 7/50
         1050/1050 [==============================] - 23s 22ms/step - loss: 0.0066 - accuracy: 0.9980 - val_loss: 0.1014 - val_accuracy: 0.9876
         Epoch 8/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0062 - accuracy: 0.9982 - val_loss: 0.0866 - val_accuracy: 0.9888
         Epoch 9/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0048 - accuracy: 0.9985 - val_loss: 0.0924 - val_accuracy: 0.9893
         Epoch 10/50
         1050/1050 [==============================] - 23s 22ms/step - loss: 0.0064 - accuracy: 0.9982 - val_loss: 0.1196 - val_accuracy: 0.9863
         Epoch 11/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0074 - accuracy: 0.9979 - val_loss: 0.0898 - val_accuracy: 0.9886
         Epoch 12/50
         1050/1050 [==============================] - 21s 20ms/step - loss: 0.0056 - accuracy: 0.9984 - val_loss: 0.1054 - val_accuracy: 0.9886
         Epoch 13/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0050 - accuracy: 0.9983 - val_loss: 0.0990 - val_accuracy: 0.9881
         Epoch 14/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0040 - accuracy: 0.9990 - val_loss: 0.0863 - val_accuracy: 0.9881
         Epoch 15/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0057 - accuracy: 0.9984 - val_loss: 0.1103 - val_accuracy: 0.9890
         Epoch 16/50
         1050/1050 [==============================] - 22s 21ms/step - loss: 0.0063 - accuracy: 0.9985 - val_loss: 0.0977 - val_accuracy: 0.9887
         Epoch 17/50
```

Test Case:

IBM Deployment space: