DIGITIAL CLASSIFICATION

Introduction

This project involves building and evaluating different models for handwritten digit recognition using both traditional machine learning algorithms and deep learning techniques. The dataset used is the MNIST dataset, which consists of 70,000 28x28 grayscale images of handwritten digits (0-9). The models implemented include K Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest Classifier (RFC), and Convolutional Neural Networks (CNN).

Requirements

- Python 3.5+
- Scikit-Learn (latest version)
- Numpy (+ mkl for Windows)
- Matplotlib

Dataset

Download the four MNIST dataset files using the following commands:

```
sh
Copy code
curl -O http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
curl -O http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
curl -O http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
curl -O http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
```

Alternatively, you can download the dataset from here, unzip the files, and place them in the respective folders.

Directory Structure

Place the dataset files in the `dataset` folder inside the `MNIST_Dataset_Loader` folder under each ML Algorithm folder:

```
markdown
Copy code
KNN
|_ MNIST_Dataset_Loader
   |_ dataset
      |_ train-images-idx3-ubyte
      |_ train-labels-idx1-ubyte
      |_ t10k-images-idx3-ubyte
      |_ t10k-labels-idx1-ubyte
```

Repeat this for the `SVM` and `RFC` folders.

Running the Code

K Nearest Neighbors (KNN)

Navigate to the KNN directory and run the `knn.py` file:

```
sh
Copy code
cd 1. K Nearest Neighbors/
python knn.py
```

Support Vector Machine (SVM)

Navigate to the SVM directory and run the `svm.py` file:

```
sh
Copy code
cd 2. SVM/
python svm.py
```

Random Forest Classifier (RFC)

Navigate to the RFC directory and run the `rfc.py` file:

```
sh
Copy code
cd 3. Random Forest Classifier/
python rfc.py
```

Convolutional Neural Networks (CNN)

Run the `CNN_MNIST.py` file directly. The MNIST dataset will be downloaded automatically:

```
sh
Copy code
python CNN_MNIST.py
```

Saving CNN Model Weights

To save the CNN model weights after training, use the following command:

```
sh
Copy code
python CNN_MNIST.py --save_model 1 --save_weights cnn_weights.hdf5
```

Loading Saved CNN Model Weights

To load the saved CNN model weights, use the following command:

```
sh
Copy code
python CNN_MNIST.py --load_model 1 --save_weights cnn_weights.hdf5
```

Accuracy

Machine Learning Algorithms

- K Nearest Neighbors: 96.67%

- SVM: 97.91%
- Random Forest Classifier: 96.82%

Deep Neural Networks

- Three Layer Convolutional Neural Network using Tensorflow: 99.70%
- Three Layer Convolutional Neural Network using Keras and Theano: 98.75%

Execution Script

An execution script is provided to run different models using command line arguments. Save the following script as `run_model.py`:

```
python
Copy code
import argparse
import os
import subprocess

BASE_DIR = os.path.dirname(os.path.abspath(__file__))

def run_knn():
    os.chdir(os.path.join(BASE_DIR, '1. K Nearest Neighbors'))
    subprocess.run(['python', 'knn.py'])

def run_svm():
    os.chdir(os.path.join(BASE_DIR, '2. SVM'))
    subprocess.run(['python', 'svm.py'])

def run_rfc():
    os.chdir(os.path.join(BASE_DIR, '3. Random Forest Classifier'))
    subprocess.run(['python', 'rfc.py'])

def run_cnn(save_model=None, save_weights=None, load_model=None):
    os.chdir(BASE_DIR)
    cmd = ['python', 'CNN_MNIST.py']
    if save_model:
        cmd += ['--save_model', str(save_model)]
    if save_weights:
        cmd += ['--save_weights', save_weights]
    if load_model:
        cmd += ['--load_model', str(load_model)]
    subprocess.run(cmd)

def main():
    parser = argparse.ArgumentParser(description='Handwritten Digit
Recognition using Machine Learning and Deep Learning')
    parser.add_argument('model', choices=['knn', 'svm', 'rfc', 'cnn'],
help='Model to run')
    parser.add_argument('--save_model', type=int, help='Save CNN model
(only for cnn)')
    parser.add_argument('--save_weights', type=str, help='File to save CNN
weights (only for cnn)')
    parser.add_argument('--load_model', type=int, help='Load CNN model
(only for cnn)')

    args = parser.parse_args()
```

```
    if args.model == 'knn':
        run_knn()
    elif args.model == 'svm':
        run_svm()
    elif args.model == 'rfc':
        run_rfc()
    elif args.model == 'cnn':
        run_cnn(args.save_model, args.save_weights, args.load_model)

if __name__ == '__main__':
    main()
```

Run the script with appropriate arguments to execute different models:

```
sh
Copy code
python run_model.py knn
python run_model.py svm
python run_model.py rfc
python run_model.py cnn --save_model 1 --save_weights cnn_weights.hdf5
```

References

The detailed paper on this project can be found at IJARCET-VOL-6-ISSUE-7-990-997.

Video Demonstration

A video demonstration of the project can be found here.

Test Images Classification Output