1 Summary

For the past few days I have been mainly setting up my local development environment and loading and exploring the data. I have installed *python 3.9.6* on my local machine. I am also using python virtual environment so that I can separate all the python packages for different use cases to different environment and have a *requirements.txt* for the ML virtual environment in case I want to move my work to a separate machine. Apart from that, I have downloaded the MNIST dataset from Kaggle (`https://www.kaggle.com/datasets/hojjatk/mnist-dataset`). There are two groups of images, Train and Test. There are total 60,000 samples for training and 10,000 samples for testing. Each image also has it's corresponding label which may not be used in our case. The images have 28x28 dimension and they are in grayscale. In order to load the data using python and explore it, I had to look into *numpy* and *matplotlib* library documentations. I created a class called *MnistDataloader* with help of some of the Jupyter Notebooks available online. I am also learning how to use Jupyter Notebooks and later I plan to use Google Colab for training and testing.

2 Samples

Below are some sample images from the DataSet.

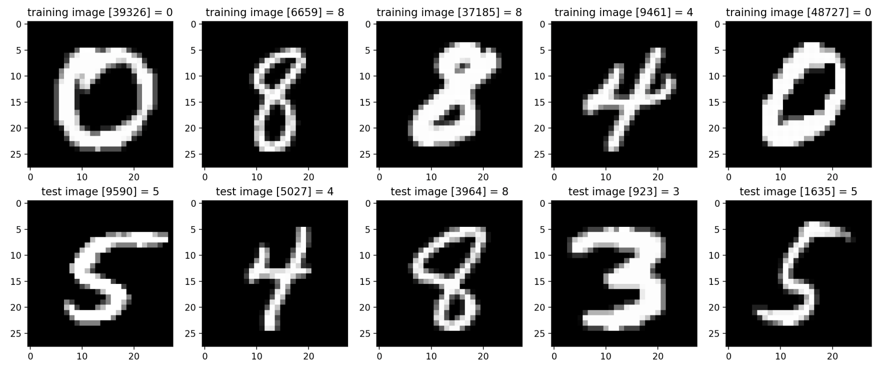
training image [39326] = 0
training image [6659] = 8
training image [37185] = 8
training image [9461] = 4
training image [48727] = 0
test image [9590] = 5
test image [5027] = 4
test image [3964] = 8
test image [923] = 3
test image [1635] = 5

Figure 1: Samples images and their labels

3 Code

I have used the data loader to load the train and test samples with their labels and I have written a function to display 5 random training samples and 5 random testing samples.

```python
import numpy as np
import struct
from array import array

#
# MNIST Data Loader Class
#
class MnistDataloader(object):
    def __init__(self, training_images_filepath, training_labels_filepath,
                 test_images_filepath, test_labels_filepath):
        self.training_images_filepath = training_images_filepath
        self.training_labels_filepath = training_labels_filepath
        self.test_images_filepath = test_images_filepath
        self.test_labels_filepath = test_labels_filepath

    def read_images_labels(self, images_filepath, labels_filepath):
        labels = []
        with open(labels_filepath, 'rb') as file:
            magic, size = struct.unpack(">II", file.read(8))
            if magic != 2049:
                raise ValueError('Magic number mismatch, expected 2049, got {}'.format(magic))
            labels = array("B", file.read())

        with open(images_filepath, 'rb') as file:
            magic, size, rows, cols = struct.unpack(">IIII", file.read(16))
            if magic != 2051:
                raise ValueError('Magic number mismatch, expected 2051, got {}'.format(magic))
            image_data = array("B", file.read())
        images = []
        for i in range(size):
            images.append([0] * rows * cols)
        for i in range(size):
            img = np.array(image_data[i * rows * cols:(i + 1) * rows * cols])
            img = img.reshape(rows, cols)
            images[i][:] = img

        return images, labels

    def load_data(self):
        x_train, y_train = self.read_images_labels(self.training_images_filepath, self.training_labels_filepath)
        x_test, y_test = self.read_images_labels(self.test_images_filepath, self.test_labels_filepath)
        return (x_train, y_train),(x_test, y_test)
```

Listing 1: loader.py

```python
import random
import matplotlib.pyplot as plt
from loader import MnistDataloader
from os.path  import join

input_path = './data'
```

```python
training_images_filepath = join(input_path, 'train-images/train-
    images-idx3-ubyte')
training_labels_filepath = join(input_path, 'train-labels/train-
    labels-idx1-ubyte')
test_images_filepath = join(input_path, 'test-images/t10k-images-
    idx3-ubyte')
test_labels_filepath = join(input_path, 'test-labels/t10k-labels-
    idx1-ubyte')

#
# Helper function to show a list of images with their relating
    titles
#
def show_images(images, title_texts):
    cols = 5
    rows = int(len(images)/cols) + 1
    plt.figure(figsize=(30,20))
    index = 1
    for x in zip(images, title_texts):
        image = x[0]
        title_text = x[1]
        plt.subplot(rows, cols, index)
        plt.imshow(image, cmap=plt.cm.gray)
        if (title_text != ''):
            plt.title(title_text, fontsize = 12);
        index += 1
    plt.show()

#
# Load MINST dataset
#
mnist_dataloader = MnistDataloader(training_images_filepath,
    training_labels_filepath, test_images_filepath,
    test_labels_filepath)
(x_train, y_train), (x_test, y_test) = mnist_dataloader.load_data()
total_train_images = len(y_train)
total_test_images = len(y_test)


#
# Show 10 random training and 5 random test images
#
images_2_show = []
titles_2_show = []
for i in range(0, 10):
    r = random.randint(1, total_train_images)
    images_2_show.append(x_train[r])
    titles_2_show.append('training image [' + str(r) + '] = ' + str
    (y_train[r]))

for i in range(0, 5):
    r = random.randint(1, total_test_images)
    images_2_show.append(x_test[r])
    titles_2_show.append('test image [' + str(r) + '] = ' + str(
    y_test[r]))

```

```
show_images(images_2_show, titles_2_show)
```

Listing 2: main.py

4 Next Steps

Since I am able to load and explore the data, I am planning to get started on GAN next. I am currently going through the GAN paper and trying to see if I can figure out implementation details from that paper. If I get stuck on GAN, I'll start implementing VAE in parallel.