Data Intake Report

Name: hello\_flask – Digital Classification using PyTorch, REST API, JSON and FLASK(Heroku part will come in later)

Report date: September 28th, 2023

Internship Batch: LISUM25

Version:<1.0>

Data intake by: Alison March

Data intake reviewer:

Data storage location: <https://github.com/alisonjing/hello_flask>

**Tabular data details:**<https://github.com/alisonjing/hello_flask/tree/master/app/data/MNIST/raw>

|  |  |
| --- | --- |
| **Total number of observations** | 60,000 training images and 10,000 testing images of handwritten digits, each image is 28 x 28( = 784) pixels. |
| **Total number of files** | 8(4 ubyte, 4 ubyte.gz) |
| **Total number of features** | 10 ( digits from 0 – 9) |
| **Base format of the file** | Ubyte.gz |
| **Size of the data** | 63.4 MB |

**Proposed Approach:**

* First, we created a new directory in the terminal called hello\_flask and set cd to hello\_flask(in Anaconda Prompt)
* Next, we created a virtual environment named venv using this command line:

conda create -n venv python=3.11.5 anaconda

* Activate the new virtual environment using activate venv
* Libraries and Packages Installation
  + flask
  + pytorch: torch and torchvision
  + waitress
  + gevent
* Create a folder and inside the folder create main.py. In this file, we created a predict function that completes the below 4 steps:
  + - Loan image
    - Convert image to tensor
    - Make a prediction.
    - Return json data
  + The following packages are imported in the main.py

**A screen shot of a computer

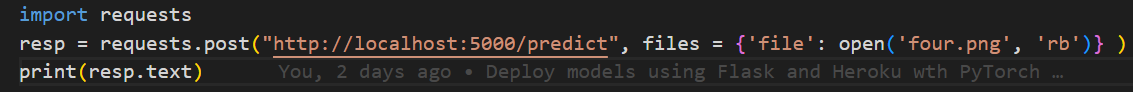
Description automatically generated**

* Testing if Flask is running once main.py is setup:
  + In the terminal(vscode) cd app
  + I am using windows system laptop, so the commands are as follows:
    - set FLASK\_APP = main.py (export FLASK\_APP =main.py for linux/macOS)
    - set FLASK\_DEBUG = development (export FLASK\_ENV = development for linux/macOS)
    - flask run

**A screen shot of a computer

Description automatically generated**

* Create a folder test and inside the test folder create a test.py

****

**\*We added in files={} part later in the resp line.**

* In the app directory, we create a new python file **torch\_utils.py**

A black screen with white text

Description automatically generated

* Next, we create a NeuralNet class and use a feedforward neural network to train and save the model, this is the load model part
* Set the Hyper-parameters as below condition:
  + **Input\_size= 784** (based on the 28 by 28 image size)
  + **Hidden\_size = 500**
  + **Num\_classes = 10 (0-9)**
  + **Model = NueralNet(input\_size, Hidden\_size, Num\_classes)**
  + **PATH = “mnist\_ffn\_pth”**
  + **Model.load\_state\_dict(torch.load(PATH))**
  + **Model.eval()**
* We convert image -> tensor: create a function called transform\_image(image\_bytes):

**A screen shot of a computer code

Description automatically generated**

**\*We used the global mean and standard deviation values to normalize the () transformation. In addition, we set the to Grayscale image transformation and output 1 channel.**

* Finally, we created a prediction () to output the predicted image in terms of value.

A screen shot of a computer code

Description automatically generated

* Add **mnist\_ffn.pth** file in the hello\_flask/app directory.
* **Main.py part:**
  + Import libraries:

**A screen shot of a computer program

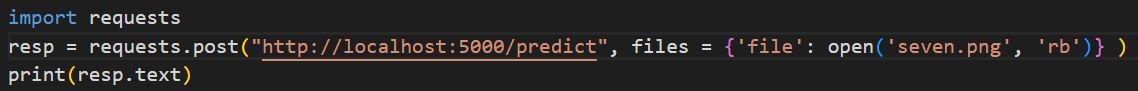
Description automatically generated**

* + **#** Create an app

**app=Flask(\_\_name\_\_)**

* + Set up debug mode
  + Define allowed image extensions if we select ‘png’, ‘jpg’, or ‘jpeg’.
  + Create a function named allowed\_file(filename) with .(dot) with maximum of 1 split, and lowercase letters mentioned in allowed image extensions.
  + Create a predict() that predicts the image of a handwritten digit.
  + Finally returns the result in jsonify().
* **A number seven in pixelated style

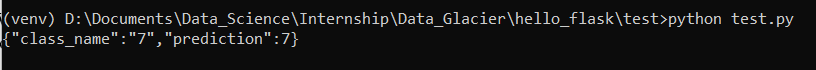
  Description automatically generated**To test if main.py and torch\_utils.py are implemented correctly, in the test.py we change the file to the actual saved image, in this case, is “seven.png”, below is the complete line of code:

****

where ‘rb’ means read binary mode.

We can test this by keeping Flask running in the background, opening an Anaconda Prompt terminal, switching to the current project directory and switching to virtual environment **venv**, and cd to test folder, then run **python test.py**:

**Result:**



**The result predicted the handwritten digit correctly (input image).**

**Note: this will not run properly if Flask is not running.**

A black screen with white text

Description automatically generatedA white cross on a black background

Description automatically generatedTo test this code again, I created an image using PAINT app and made a handwritten four “4”(in 28 by 28-pixel size with black canvas and white paintbrush color) and saved it as four.png before updating the image in the test.py.

* Create a file wsgi.py and input the following line: from app.main import app
* We then create a **Procfile** file for Heroku and input the following line: Web: waitress-serve wsgi:app
* Create a **runtime.txt** with python version: python-3.11.5