

This assignment does not count toward the final grade.

Opportunity 2: Build a ML Classifier

Due Mar 20 by 11:59pm **Points** 50 **Submitting** a file upload **File Types** zip
Available Feb 15 at 8am - Mar 20 at 11:59pm

This assignment was locked Mar 20 at 11:59pm.

MNIST Handwritten Digit Recognition



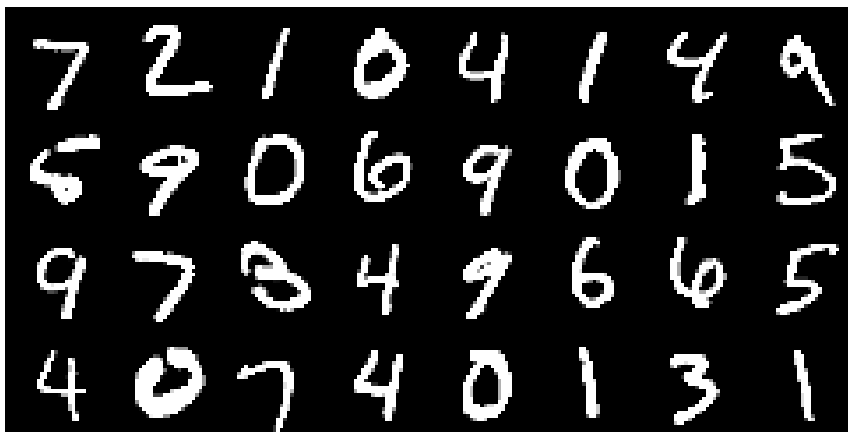
You'll write a script that trains/tests a machine learning (ML) model for handwritten digit classification in **PyTorch** (<https://pytorch.org/>).

Learning Outcomes

- Write a script for training and testing a ML classifier (or a ML model)
- Train a ML classifier and test the model's performance
- (Optional) Explore the impact of training hyper-parameters on the model's performance.

Handwritten Digit Recognition

You will train a ML classifier that performs handwritten digit classification. You will use the **MNIST database** (https://en.wikipedia.org/wiki/MNIST_database) (or the MNIST dataset) containing 60k training and 10k testing 28x28 grey-scale images of handwritten digits (from zero to ten). It is a standard database used for benchmarking various machine-learning algorithms and techniques. Here, for your understanding of how the data looks like, are 32 example images taken from the database, 8 images per row.



Building a ML Classifier

Your objective is to train your ML classifier (or your ML model) on the MNIST database (on the 60k training data) and achieve a classification accuracy of over 95% on the 10k testing data. Typically, we perform the training using mini-batch SGD as follows:


1. Draw a batch of samples *randomly* from the training data
2. Make your model predict the labels of these samples
3. Compute the error value (i.e., loss) of the model's predictions
4. Update the model to make correct predictions over a new batch
5. Perform 1-4 steps iteratively until the error value (i.e., loss) is sufficiently small


At each iteration, we perform the testing of a classifier (or a model) on the 10k test-time samples (i.e., the testing data) and check if the model achieves sufficient accuracy. Once the accuracy at any iteration is greater than the previous model's performance, we will store the model as **mnist_model.pth** file. If there's a model file that already exists, we will overwrite it.

Instructions

Here we provide you with a skeleton code written in Python that trains (or tests) a ML classifier.

Initial Setup

The skeleton code uses a popular deep-learning library, **PyTorch**  (<https://pytorch.org/>). To run the skeleton code, you need to install both Python and Pytorch on your environment. Please take the following steps. **Note that you don't need to run this assignment on OS1 server and also recommend to use Python 3.9.**

```
$ python3 --version           // Check if you have python3.9
Python 3.9.13
$ pip3 install torch torchvision // Install PyTorch (this is for Mac; see https://pytorch.org/
 https://pytorch.org/ for other environments)
```

If you want to use Python virtual environment to minimize the collision between the Python in your system, you can use python virtual env:

```
$ python3 -m venv <your environment name, e.g., ml>
$ source <your env. name>/bin/activate           // need to do when you open your shell
(<your env. name>)$                               // make sure the environment name appears before $,
otherwise run the above "source ..." command again.
```

Skeleton Code


```

: [test:epoch:1]: 100%|████████████████████████████████████████████████████████████████████████████████| 313/313 [00:0
3<00:00, 82.69it/s]
: Train loss/acc. [0.02 / 86.61%] | Test loss/acc. [0.26 / 92.49%]
: Store the best model [0.00 -> 92.49]
: [train:epoch:2]: 100%|████████████████████████████████████████████████████████████████████████████████| 1875/1875 [00:4
8<00:00, 38.81it/s]
: [test:epoch:2]: 100%|████████████████████████████████████████████████████████████████████████████████| 313/313 [00:0
3<00:00, 81.62it/s]
: Train loss/acc. [0.01 / 93.22%] | Test loss/acc. [0.19 / 94.26%]
: Store the best model [92.49 -> 94.26]

... (training) ...

: [train:epoch:10]: 100%|████████████████████████████████████████████████████████████████████████████████| 1875/1875 [0
0:46<00:00, 39.99it/s]
: [test:epoch:10]: 100%|████████████████████████████████████████████████████████████████████████████████| 313/313 [0
0:03<00:00, 86.80it/s]
: Train loss/acc. [0.00 / 98.19%] | Test loss/acc. [0.06 / 98.09%]
: Store the best model [97.94 -> 98.09]

```

Testing a model.

```

$ python train.py --test --model mnist_model.pth
{
  "num_workers": 4,
  "model": "mnist_model.pth",
  "batch_size": 32,
  "epoch": 10,
  "lr": 0.01,
  "test": true
}
: [test:epoch:n/a]: 100%|████████████████████████████████████████████████████████████████████████████████| 313/313 [0
0:03<00:00, 84.40it/s]
: Test loss/acc. [0.06 / 98.09%]

```

Hints

- You can refer to some example code or tutorials on the Internet [[link](https://github.com/pytorch/examples/tree/main/mnist)  (<https://github.com/pytorch/examples/tree/main/mnist>)]

What to turn in?

- Required:** Upload a zip file that contains the constructed model file **mnist_model.pth** on Canvas

Grading Criteria

- This assignment is worth 2% of your final grade.
- The grading can be done on any machine (**No need to be on OS1**)

- Sanghyun will test if:
 - Your model can achieve the test-time accuracy above 95%