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



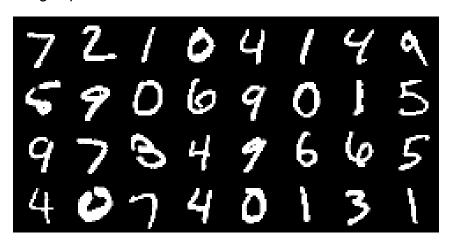
You'll write a script that trains/tests a machine learning (ML) model for handwritten digit classification in Pytorch (https://pytorch.org/).

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 (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.

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

Skeleton Code

Here is a description of the <u>sample script</u>

(https://canvas.oregonstate.edu/courses/1901760/files/97634319?wrap=1) ↓
(https://canvas.oregonstate.edu/courses/1901760/files/97634319/download?download_frd=1) we provide.
You can find if __name__ == "__main__": and that's the starting point.

- 1. Get the command line arguments; the script currently accepts 6 arguments:
 - 1. --num-workers: how many processes do you want to use to load the training data from storage.
 - 2. --model: the location (path) to the ML classifier file (only required for testing)
 - 3. --batch-size: how many data samples you want to show at once to your ML model <u>during</u> <u>training</u>
 - 4. --epoch: how many number of iterations you want to run during training
 - --Ir: how drastically you want your model to "correct" its behaviors <u>during training</u>.
 If we use a large learning rate (Ir), it can prevent your model from learning anything.
- 2. Run training / testing run_traintest(args)
 - 1. It loads the MNIST dataset (the script will automatically download it from the Internet)
 - 2. It initializes a ML model (**SimpleNet** a convolutional neural network with 2 convolutional and 2 fully-connected layers)
 - 3. It initializes an optimizer (the optimizer will be used to minimize the error between the model's prediction and the true labels).
 - 4. (In the training mode) It runs training of the model and store the model to mnist_model.pth if it's accuracy is higher than before
 - 5. (In the testing mode) It runs predictions on the test data and report the accuracy.

You have to modify the script to enable training or testing a ML classifier. The code locations where you need to modify are marked as **TODO** -. Once you complete the modification, you need to run the script and get the stored model (**mnist_model.pth**). If you want to check the performance of your trained model, you can run the testing command in the example outputs.

Example Outputs

Here are example executions of the program.

Training a model

```
$ python train.py
{
    "num_workers": 4,
    "model": "",
    "batch_size": 32,
    "epoch": 10,
    "lr": 0.01,
    "test": false
}
    : [train:epoch:1]: 100%|
7<00:00, 39.45it/s]</pre>
1875/1875 [00:4
```

```
: [test:epoch:1]: 100%
                                                                               113/313 [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%||
                                                                                  II 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%|
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.

Hints

You can refer to some example code or tutorials on the Internet [link => (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%