This assignment does not count toward the final grade.

# Opportunity 3: Multi-process Data Loader

**Due** Mar 20 by 11:59pm **Points** 75 **Submitting** a file upload **File Types** txt
**Available** Feb 20 at 8am - Mar 20 at 11:59pm

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

## Multi-process Data Loader

Here, we will compare the time it takes to load data with multiple processes. Note that this assignment requires the code from the assignment "Opportunity 2: Build a ML Classifier." Please complete that assignment first before doing this one.

### Learning Outcomes

- Learn several ways to measure the running time of a process.
- Measure the time it takes to load data with multiple processes.

## Instructions

You should use your Python code written for the assignment "Opportunity 2: Build a ML Classifier." If you haven't finished the assignment and are looking at this instruction, press the "go back" button in your browser and finish the "Opportunity 2" assignment. **Note that you don't need to run this assignment on OS1 server; I advise to run on your own server not to disturb others working on the OS1 server.**

**Task 1: Measure the wall-time**

The skeleton code that we provide in "Opportunity 2" receives a command line argument that you can control the number of processes the script will use for loading data. You can do it by simply adding `--num-workers <# of processes>` We will run our script with a single process, two processes, and four processes.

Now run the following set of commands three times for each case (1-/2-/4-processes) and store the time it takes to run.

```
$ time python train.py --num-workers 1
... (you will see the training outputs)

real    9m1.034s
user    14m40.498s
sys     3m27.195s
```

The `time` command allows you to measure the time it takes to complete the running. Compute the average time it takes to run for each of the 1-process, 2-process, and 4-process cases over three runs. You need to use the `real` time from the output of the `time` command.

**Task 2: Measure the time from Python**

Now, let's use the time package supported by Python. You need to make the following changes to your `train.py` script.

(1) Include the time package.

```
"""
import time    // add this line
import json
```

(2) Record the start time.

```
...
    train_records = []

    # : record the start time    // add this line
    start_time = time.time()     // add this line
...
```

(3) Measure the time it takes to finish the training.

```
...
        _best_acc = test_acc

    elapsed = time.time() - start_time
    print (' : Time elapsed to train a model {:.2f} seconds'.format(elapsed))
...
```

Now run training three times for each case (1-/2-/4-processes) and store the time it takes to run. Compute the average time it takes to run for each of the 1-process, 2-process, and 4-process cases over three runs.

## What to turn in?

- **Required:** Upload a txt file (**time.txt**) to Canvas containing the following info.

```
[Time measured by the time command]
1-process: TODO (seconds) on avg.
2-process: TODO (seconds) on avg.
4-process: TODO (seconds) on avg.

[Time measured by the Python time()]
1-process: TODO (seconds) on avg.
2-process: TODO (seconds) on avg.
4-process: TODO (seconds) on avg.
```

## Grading Criteria

- This assignment is worth 3% of your final grade.
- The grading can be done on any machine (**No need to be on OS1**)