Project: Efficiency of Our Matlab/Octave Implementation
Goal

- Timing comparison between our code and PyTorch CNN
We have had our own implementation
From the discussion, we think ours may be as efficient
It’s time to check if that’s the case
Let’s run
- PyTorch’s SG for 5 epochs
- Our SG for 5 epochs
Check and analyze the running time per epoch
To make sure that they have the same amount of operations, we do the simplest SG (no momentum and anything else)

You need to use the same network architecture (see below)

We also use the same mini-batch size 128

However, no need to worry if they use the same initial solution (as accuracy isn’t important now)

Nor do we worry about their pre-processing steps. These things shouldn’t affect the input size and therefore the amount of computation.
A key thing to check is the percentage of each main operation of our implementation (see the list of operations in our slides).

To do this, based on materials in our lectures you want to trace the code and know details.

Thus you need to do MATLAB/Octave profiling.
Using Our Implementation I

- About how to run our implementation, check the main github page (i.e., README.md) in detail.
- You must put two configuration files in the config sub-directory.
- You also need a driver file.
- We give these files on the course web page.
- You may need to modify the driver file for your experimental need.
- For your convenience, we also provide data in MATLAB/Octave mat format in the same directory.
Using Our Implementation II

- However, I didn’t check if the data sets are exactly the same as what we used when doing PyTorch experiments. Someone please help to do it.
- Any comments to the use of the implementation are welcome.
Let’s use only one core now

For MATLAB, the following command specifies that one core is used

```
matlab -singleCompThread
```

For octave, we can use

```
export OMP_NUM_THREADS=1
```

For PyTorch, we also do

```
export OMP_NUM_THREADS=1
as torch.set_num_threads(1) does not seem to work.
```
An issue of the above setting is that PyTorch runs 2 threads and uses 50% CPU on each. We haven’t found a way to force it to use one core. Thus the comparison may not be completely fair. You can see if you have better settings.
Because we use the \texttt{randsample} command to select a subset for gradient evaluation, you need to install Octave statistics toolbox.

For example, on Ubuntu, you need

\% sudo apt-get install octave-statistics
How to know which optimized BLAS used by MATLAB/Octave?

You can do

```
octave:4> version(’-blas’)
an = OpenBLAS (config: NO_LAPACKE DYNAMIC_ARCH...)
```

You may try to build Octave by linking Intel MKL