# Project: Efficiency of Our Matlab/Octave Implementation

Last updated: May 25, 2020

# Goal

- Running time analysis of our SG implementation in MATLAB and how efficient it is in comparison with Tensorflow
- MATLAB: we explicitly conduct every operation discussed in slides
- Tensorflow: it calculates the gradient for us.

# Project Contents I

- In simpleNN there is a MATLAB implementation
- We are interested in checking how efficient it is
- Let's run
  - Tensorflow's SG for 10 epochs
  - Our SG for 10 epochs
- Check and analyze the running time per epoch
- Let's use the simple SG without momentum (so not Adam). That is, the simplest setting
- We use the same `CNN_4layers` architecture as before

# Project Contents II

- Ensure that they use the same parameters (e.g., batch size)
- However, no need to worry if they use the same initial solution (as accuracy isn't important now)
- They do the same pre-processing steps though at this moment you don't worry about what these steps are

  These things shouldn't affect the input size and therefore the amount of computation

# Project Contents III

- 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
- To see time of each operation or each subroutine, you must do MATLAB/Octave profiling
- Another thing to check is the timing comparsison with Tensorflow.

# Project Contents IV

Separately analyzing main operations in Tensorflow may not be easy (?) so let's focus only on the total training time

- For both MATLAB and Tensorflow, make sure you check only the training time (i.e., no validation)
- For this project let's still consider the same data sets as before
- This project is more research oriented than the earlier ones.

For the final grading the weight of this project may be higher

# Project Contents V

- Make sure you run both simpleNN and Tensorflow on CPU without using GPU

  Otherwise your timing analysis may be incorrect

# Using the MATLAB Implementation I

- All details are given in README
- You must put two configuration files for the two data sets in the `config` sub-directory
- You also need a driver file
- We give a sample driver file called `example.m` but you may modify the driver file for your need
- Remember to specify `-gpu_use 0` for using CPU only

# About `example.m` I

- You should trace the code because it handles some tricky things
- For example,

  ```
  Z = [full(Z) zeros(size(Z,1),
       a*b*d - size(Z,2))];
  ```

  gives zeros columns in the end.

  If the input matrix is in the sparse format, zero columns in the end are not stored.
- You don't need to understand details of the two normalization steps in the code

# Issue of Multiple Cores I

- Both Tensorflow and MATLAB try to use multiple cores
- But for MATLAB we know only some of the main operations use multi-core
- So the timing comparison can be tricky
- For MATLAB, let's focus on getting correct wall-clock time of each major operation
- You can check both single and multi-core settings
- For MATLAB, the following command specifies that one core is used

# Issue of Multiple Cores II

```
matlab -singleCompThread
```

- For octave, we can use

```
export OMP_NUM_THREADS=1
```

- For Tensorflow let's just use the default (multi-core)

# Single Versus Double I

- By default Tensorflow uses single
- Let's use single in MATLAB too
- You can use the option `-ftype` to specify the use of single

# Octave I

- Octave's profiling funcationality is not as good as Matlab's yet
- It may not show the time spent on each line
- However, from the time of each function call, you should still be able to do some analysis

# Optimized BLAS I

- How to know which optimized BLAS used by MATLAB/Octave?
- You can do

```
octave:4> version('-blas')
ans = OpenBLAS (config: NO_LAPACKE DYNAMIC_A
```

- You may try to build Octave by linking Intel MKL
- You can follow the procedure in the section `Link/Build Latest Octave with latest MKL` at `https://software.intel.com/en-us/articles/using-intel-mkl-in-gnu-octave`

# Optimized BLAS II

- you may need to add

  `--enable-fortran-calling-convention=gfortran`

  into the configure options to build Octave.

# Presentation

- Students with the following IDs:

  t08901107

  r07922154

  b05902105

  t08902130

  b05201037

  b06902008

  b05902050

  a08922203

  r08521508

  please do a 10-minute presentation (9-minute the contents and 1-minute Q&A)

# Acknowledgments


- Pin-Yen Lin helped to figure out many settings described in this file
- Chien-Chih Wang helped to check the driver file