**Winter 2019**

# CS 133 Lab 4

### GPU w/ OpenCL: Convolutional Neural Network (CNN)

## Description

Your task is to accelerate the computation of convolutional neural network (CNN) using OpenCL with a GPU. The details of the algorithm can be found in the lecture slides and the Lab 3 description.

- We will use `g3s.xlarge` instances for grading.

## Preparation

### Create an AWS Instance

Please refer to the tutorial slides and create an `g3s.xlarge` instance with a Ubuntu 18.04 AMI.

Please use your **AWS account** (not AWS Educate classroom account) for this lab. You will probably need to request increase of limit for this type of instance. To do this, go the AWS console -> Limits -> “Request limit increase” next to `g3s.xlarge`. Please use US East (N. Virginia).

When launching the instance, please click “storage” and change the size to 15GB. The cuda installation process takes more than the default 8GB of disk space and fails if unchanged.

### Run CNN with OpenCL

We have prepared the host code for you at GitHub. The files are mostly the same as Lab 3 except changes to the makefile that enables GPU. Also, the kernel file has renamed to `nvidia.cl`, and a new setup file for gpu has been added : `gpu_setup.sh`. Log in to your instance and run the following commands:

```
git clone https://github.com/UCLA-VAST/cs-133-19w -o upstream
cd cs-133-19w/lab4
./gpu_setup.sh
make test
```

The provided code will load test data and verify your results against a ground truth. It should run with large error and finish in a few seconds.

#### Tips

- To resume a session in case you lose your connection, you can run `screen` after login.
- You can recover your session with `screen -DRR` if you lost your ssh connection.
- You should ***stop*** your instance if you are going back and resume your work in a few hours or days. Your data will be preserved but you will be charged for the EBS storage for $0.10 per GB per month (with default settings).
- Using instance type `g3s.xlarge` costs $0.75 per hour. Please pay careful attention to your spendings.

- You should ***terminate*** your instance if you are not going to back and resume your work in days or weeks. ***Data on the instance will be lost***.
- You are recommended to use ***private*** repos provided by GitHub. ***Do not put your code in a public repo.***

Your task is to implement a fast, parallel version of CNN on GPU. You can start with the sequential version provided in `cnn.cpp`. You should edit `nvidia.cl` for this task. To adjust the global / local work size parameters, edit `params.sh`. For example, if you would like to set the global work size to be (1, 1, 1), you should uncomment the second line of `params.sh` by deleting the leading pound sign (#). Note that the work size does not necessary have to be 3-dimensional (can be just 1 dimension). You cannot put spaces around the equal sign (=) in `params.sh`. If your work size is multi-dimensional, you cannot omit the quote marks (').

### Tips

- To check in your code to a ***private*** GitHub repo, create a repo first.

```
git branch -m upstream
git checkout -b master
git add nvidia.cl params.sh
git commit -m "lab4: first version"  # change commit message accordingly
# please replace the URL with your own URL
git remote add origin git@github.com:YourGitHubUserName/your-repo-name.git
git push -u origin master
```

- You are recommended to `git add` and `git commit` often so that you can keep track of the history and revert whenever necessary.
- If you move to a new instance, just `git clone` your repo.
- Run `make test` to re-compile and test your code.
- You can run the sequential CNN by `make test-seq`.
- If `make test` fails, it means your code produces wrong result.
- ***Make sure your code produces correct results!***

# Submission

You need to report your performance results of your GPU-based OpenCL implementation on an `g3s.xlarge` instance. Please express your performance in GFlops and the speedup compared with the sequential version in CPU. In particular, you need to submit a brief report which summarizes:

- Please explain the parallelization strategies you applied for each step (convolution, max pooling, etc) in this lab. How does it differ from your Lab 3 CPU parallelization strategy? What made you change your strategy?
- Please describe any optimization you have applied. (*Optional, bonus +8(=2x4)*: Evaluate the performance of at least 4 different optimization techniques that you have incrementally applied and explain why such optimization improves the performance. Simply changing parameters does not count and sufficient code change is needed between versions. In your report, please include the most important changes you have applied to your code for each optimization.)
- Please report the number of work-groups (NOT global work size) and work-items per work-group that provides the best performance for your kernel. Then please look up the number of multiprocessors and CUDA cores per multiprocessor in the GPU of g3s.xlarge. Do the numbers match? If not, please discuss the probable reason. (*Optional, bonus +2*: Please

include a table that shows the performance for different number of work-group and work-items per work group. Please report the performance for at least 3x3=9 different configurations including the one that provides the best performance. The spacing between different work-group/work-item should be around 2X different - e.g. #work-group- 8,16,32. )

- *Optional*: The challenges you faced, and how you overcame them.

You will need to submit your optimized kernel code and the parameter settings. Please do not modify or submit the host code. Please submit to CCLE. Please verify the ***correctness*** of your code before submission.

Your final submission should be a tarball which contains and only contains the following files:

```
<Your UID>.tar.gz
 └ <Your UID>
   ├ nvidia.cl
   ├ params.sh
   └ lab4-report.pdf
```

File `lab4-report.pdf` must be in PDF format. You should make the tarball by copying your `lab4-report.pdf` to the `lab4` directory and running `make tar UID=<Your UID>`. If you made the tarball in other ways, you ***MUST*** put it in the `lab4` directory and check by running `make check UID=<Your UID>`.

# Grading Policy

## Submission Format

**Your submission will only be graded if it complies with the requirement. In case of missing reports, missing codes, or compilation error, you will receive 0 for the corresponding category/categories.**

## Correctness (50%)

Please check the correctness of your implementation.

## Performance (25%)

Your performance will be evaluated based on the workgroup settings you set in `params.sh`. The performance point will be added only if you have the correct result, so please prioritize the correctness over performance. Your performance will be evaluated based on the ranges of throughput (GFlops). We will set five ranges after evaluating all submissions and assign the points as follows:

- Better than TA's performance: 25 points + 5 points (bonus)
- Range A GFlops: 25 points
- Range B GFlops: 20 points
- Range C GFlops: 15 points
- Range D GFlops: 10 points
- Speed up lower than range D: 5 points
- Slowdown: 0 points

## Report (25%)

Points may be deducted if your report misses any of the sections described above.