Spring 2025

CS 133 Lab 3

GPU w/ CUDA: Convolutional Neural Network (CNN)

Due Date: 5/15/2025 02:00pm

Description

Your task is to accelerate the computation of two layers in a convolutional neural network (CNN) using CUDA on a GPU. Specifically, for an input image with 228×228 pixels and 256 channels, you are going to:

1) Implement a 2D convolutional layer with 256 filters of shape [256, 5, 5]
2) Apply ReLU activation `relu(x) = max{x, 0}` with bias values for each output channel
3) Implement a 2D max-pooling layer with 2×2 window

In the `cnn_gpu.cu` file, you will need to implement the above as a CUDA device kernel declared as:

```
__global__ void cnn_gpu(float* input, float* weight, float* bias, float* output)
```

Where `input` is the input image of size [256][228][228], `weight` is the convolution filters of size [256][256][5][5], `bias` is the offset values of size `[256]` that will be added element-wise to each output channel, and `output` is where you write the results `maxpool(relu(conv2d(input, weight) + bias))`. The output size is [256][112][112].

Figure 1 shows the CNN layers in this lab. You can also refer to the discussion and lecture slides for further intuition of the workflow.

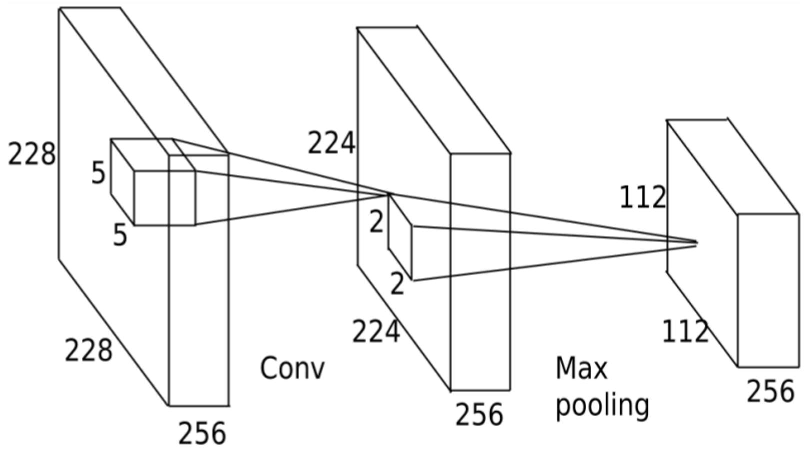


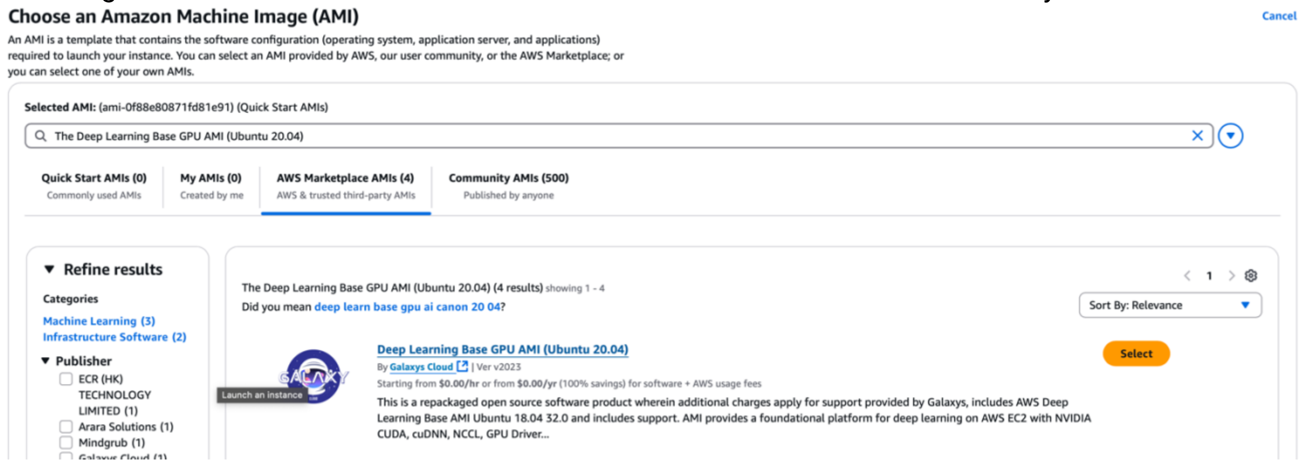
Figure 1: The CNN layers to be accelerated

How-To

Create an AWS Instance for Development

You may be able to first develop code locally and even compile CUDA code without a GPU on your personal computer. However, you will need AWS GPU instances to check your correctness and performance. Beware, getting a CUDA development environment setup can be very time-consuming, **move to your AWS instance if you have non-trivial issues**.

Please refer to the discussion slides to create a **`g4dn.xlarge`** GPU instance with **Deep Learning Base GPU AMI (Ubuntu 20.04)**. Be careful when using your AWS account because if you use up all your credits, you may end up paying the expenses yourself. If you use fewer credits, you may have some remaining after the course to be used at your discretion.



Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Selected AMI: (ami-0f88e80871fd81e91) (Quick Start AMIs)

The Deep Learning Base GPU AMI (Ubuntu 20.04)

The Deep Learning Base GPU AMI (Ubuntu 20.04) (4 results) showing 1 - 4

Did you mean deep learn base gpu ai canon 20 04?

Deep Learning Base GPU AMI (Ubuntu 20.04)

By Galaxys Cloud | Ver v2023

Starting from $0.00/hr or from $0.00/yr (100% savings) for software + AWS usage fees

This is a repackaged open source software product wherein additional charges apply for support provided by Galaxys, includes AWS Deep Learning Base AMI Ubuntu 18.04 32.0 and includes support. AMI provides a foundational platform for deep learning on AWS EC2 with NVIDIA CUDA, cuDNN, NCCL, GPU Driver...

IMPORTANT: During the launching process of the GPU instance, please use the default "Storage" as this lab requires significant space. Starting the instance will fail otherwise.

We have prepared the starter kit for you on GitHub. Log in to your AWS instance and run:

```
unzip lab3.zip
cd lab3
nvcc --version # should show 11.8
```

Test CUDA Example on GPU: Vector Add

We have prepared a simple CUDA program for vector add as reference for CUDA programming. You can review it in `lib/vadd.cu`. To test it, run:

```
make test-vadd
```

It will be built and run with the parameters specified in `params.sh`. If you see "PASS" in the output, the VADD kernel has been correctly built and executed.

Test Sequential CNN on CPU

We have provided a sequential CNN implementation on CPU as reference for the kernel behavior. You can review it in `lib/cnn_seq.cu`. To test it, run:

```
make test-seq
```

You should see that it passes the check, as well as its performance.

```
Time: 12.41 s
Perf: 13.2487 Gflops
PASS
```

Develop Parallel CNN with CUDA on GPU

You can now start to develop your parallel CUDA CNN kernel. The provided starter kit will load the test data from binary files and verify your results with the ground truth data.

Your task is to implement a fast, parallel version of the CNN kernel we defined on GPU in `cnn_gpu.cu`. You can start by adapting the sequential CNN kernel into CUDA device code. You can use the macros defined in `lib/cnn.cuh`. Your own definitions must be in `cnn_gpu.cu`.

To adjust the grid size parameters, edit `params.sh`. For example, if you would like to set the grid size to be (4, 2, 1), you should change the line in `params.sh` to `export GRID ='4 2 1'`, listed in x, y, z order. The sizes you choose don't necessarily have to be 3-dimensional because you can set dimensions that you don't want to be 1. However, leveraging sizes in 3-dimensional can make it easier to reason about the parallelism and also simplify indexing in your CUDA kernel. The format in `params.sh` is strict: you cannot put spaces around the equal sign (=). Also, you cannot omit the quote marks (').

To test your implementation of CNN kernel on GPU:

```
make
```

If you see something similar to the following message, your implementation is incorrect.

```
Found 2120190 errors
FAIL
```

If you see "PASS", your implementation is correct. Your **performance** will be shown after `Perf`.

⚠ **Note**: Use of Tensor Cores is **not allowed** in this lab. You must implement your kernel using standard CUDA cores with float data types only. Do not use __half, WMMA APIs, or any Tensor Core instructions.

Tips

- **We will use `g4dn.xlarge` instances for grading.**
- When you develop on AWS:

- **`g4dn.xlarge` costs $0.53 per hour. Please pay careful attention to your spending.**
- **Deep Learning Base GPU AMI (Ubuntu 20.04)** cost $0.80 (one time fee not covered by the credit). The Deep Learning Base GPU AMI (Ubuntu 20.04) incurs a one-time fee of $0.80, which is not covered by the provided credit.
- To resume a session in case you lose your connection, you can run `screen` after login. You can recover your session with `screen -DRR`. Alternatively, you can use `tmux`. You should ***stop*** your AWS instance if you will come 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 at $0.10 per GB per month (with default settings). You should ***terminate*** your instance if you will not come back and resume your work. ***Data on the instance will be lost***.

- You are recommended to use ***private*** repositories provided by GitHub to backup your code. ***Never put your code in a public repo to avoid potential plagiarism.*** To check in your code to a private GitHub repo, create a repo first.

```
git branch -m upstream
git checkout -b main
# your modifications
git commit -m "lab3: first commit" # 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 main
```

- You are recommended to `git add` and `git commit` often so that you can keep track of the history and revert whenever necessary.
- **You can perform multiple executions and choose the maximum value for the performance in your reports.**
- ***Make sure your code produces correct results!***

Submission

You need to report the performance of your CUDA implementation with an NVIDIA GPU on a `g4dn.xlarge` instance. Please report your performance in GFlops and **present any comparison you make in tables.** Your report should include the following aspects:

- (9pts) Please explain the **parallelization strategies** (how data is distributed and how computation is done) you applied for each loop in the CNN program (convolution, max pooling, etc) in this lab. What made you choose this strategy and how is it justified by the specs of our GPU device?
- (9pts) Please describe **any optimization** you have applied.
- (7pts) Please report the dimension of threads per block and blocks per grid that provides the best performance for your kernel. Then please look up the number of multiprocessors and CUDA cores per multiprocessor in the T4 GPU. **Does the GPU specification corroborate your chosen dimensions**? If not, please discuss the probable reason.
- *Optional*: The challenges you faced, and how you overcame them.
- ***(Bonus +8pts)***: Please evaluate the performance of at least four (4) different optimization techniques that you have incrementally applied and explain why such optimization improves the performance. Changing parameters does not count as one optimization. Significant code

changes are needed between versions to be counted. In your report, please include the most important changes in your code for each optimization. Git commits, branches or tags make this process easy.

- **(*Bonus +2pts)***: Please include a table that shows the performance for different threads per block and blocks per grid. Please report the performance for at least 3x3=9 different configurations including the one that provides the best performance. Your dimensions of thread blocks and the grid should probably be factors of loop bounds or powers of 2.

You also need to submit your optimized kernel code and the best-performing parameter settings. Do not modify the host code in the `lib` directory. Please submit on Gradescope. You will be prompted to enter a Leaderboard name when submitting, please enter the name you want to show to your classmates. You can submit as many times as you want before the deadline to improve your performance. Your final submission should contain and only contain these files individually:

```
├ cnn_gpu.cu
├ params.sh
├ run.log
└ lab3-report.pdf
```

File `lab3-report.pdf` must be in PDF format exported by any software.

To create the zip, please run:

```
make zip
```

Grading Policy

Your submission will be automatically graded on submission format, correctness and performance. The last two aspects are tested on the same experiment environment mentioned above (`g4dn.xlarge`). **Your last correctness and performance marks will be final** if there are no cheating or failed runs. You can see the performance of other students on the Leaderboard. You could request a manual regrade if there is a significant discrepancy between the performance on your instance and Gradescope.

Your submission will only be graded if it complies with the formatting requirements. In the case of missing reports/code, or compilation errors, you will receive 0 for the corresponding category(ies). Please fix your files and resubmit if the autograder returns you an error.

Correctness (50%)

Please check the correctness. You can see this score on Gradescope soon after your submission.

Performance (25%)

Your performance will be evaluated using the grid size settings you set in `params.sh`. The performance points 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). Ranges A+ and A++ will be decided after all the submissions are graded:

- Range A++, better than Range A+ performance: 25 points + 20 points (bonus)
- Range A+, better than Range A performance: 25 points + 10 points (bonus)
- Range A GFlops [460, 860]: 25 points
- Range B GFlops [260, 460): 20 points
- Range C GFlops [160, 260): 15 points
- Range D GFlops [60, 160): 10 points
- Range E GFlops [4, 60): 5 points
- Lower than range E [0, 4): 0 points

Report (25%)

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

Academic Integrity

All work is to be done individually, and any sources of help are to be explicitly cited. Any instance of academic dishonesty will be promptly reported to the Office of the Dean of Students. Academic dishonesty includes, but is not limited to, cheating, fabrication, plagiarism, copying code from other students or from the internet, or facilitating academic misconduct. We'll use automated software to identify similar sections between **different student programming assignments**, against **previous students' code**, or against **Internet sources**. Students are not allowed to post the lab solutions on public websites (including GitHub). Please note that any version of your submission must be your own work and will be compared with sources for plagiarism detection.

Late policy: Late submissions will be accepted within **24 hours** with a 10% penalty. No late submission will be accepted after that (you will lose all points after the late submission time).