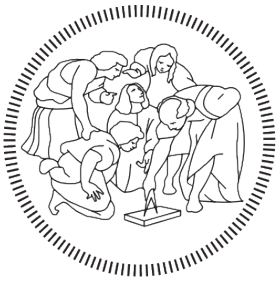

POLITECNICO
MILANO 1863

POLITECNICO DI MILANO

MASTER'S PROGRAM IN
HIGH PERFORMANCE COMPUTING ENGINEERING

REPORT
2D-Convolution Algorithm in CUDA

Report prepared by:
Salvatore Mariano Librici
Person Code: 11078653

Rong Huang
Person Code: 10948935

Academic Year:
2024/2025

Report

2D convolution is a fundamental operation in image processing and deep learning. It involves applying a kernel (or mask) over a matrix to extract features or perform transformations. Due to the computational intensity of this operation, leveraging GPUs for acceleration is crucial.

In this study, we implemented two CUDA-based approaches for 2D convolution:

- Shared memory without tiling.
- Shared memory with tiling.

The experiments were conducted on a Tesla T4 GPU with the following specifications:

- 40 Streaming Multiprocessors (SMs).
- 49152 bytes of shared memory per block.
- 320 GB/s peak memory bandwidth.

Methodology

The CUDA implementation uses dynamic shared memory to load portions of the matrix and kernel into faster shared memory. The two methods are:

1. **No Tiling:** Entire input and kernel loaded into shared memory without subdivision.
2. **With Tiling:** Matrix is subdivided into tiles, and shared memory is reused efficiently.

The experiments involved the following parameters:

- **Matrix sizes:** 32×32 to 10000×10000.
- **Kernel sizes:** 3, 7, 15, 31, 63, 127.
- Execution time measured using CUDA events.

Shared Memory Utilization

The CUDA implementation leveraged shared memory to accelerate data access during convolution. The shared memory usage was carefully designed to accommodate both the input matrix tiles and the kernel:

- **No-Tiling Approach:** The entire input matrix and kernel were loaded into shared memory. This resulted in higher shared memory usage, especially for larger matrices and kernels, potentially exceeding the shared memory limit of 49152 bytes per block.
- **Tiling Approach:** The matrix was divided into smaller tiles, each fitting within shared memory along with the kernel. This method improved shared memory utilization by reusing the loaded data for multiple computations, significantly reducing redundant global memory accesses.

The shared memory requirements were calculated to ensure the data fit within the available memory for the Tesla T4 GPU. For the tiling approach, the shared memory size was computed as:

$$\text{Shared Memory Size} = (\text{Tile Width} + \text{Kernel Width} - 1)^2 + (\text{Kernel Width})^2$$

This configuration maximized performance while staying within the hardware limits, particularly for kernel sizes up to 127×127. For larger kernels, additional optimizations, such as partitioning the kernel or using global memory more effectively, would be necessary.

Results

Performance Comparison: Table 1 summarizes the execution times for the two approaches across different matrix and kernel sizes.

Table 1: Performance comparison of 2D convolution methods.

Matrix Size	Kernel Size	No Tiling (ms)	With Tiling (ms)
32	3	0.231200	0.049120
32	7	0.020256	0.018144
32	15	0.029408	0.026624
32	31	0.067584	0.066944
128	3	0.015968	0.014208
128	7	0.020512	0.018464
128	15	0.040160	0.038560
128	31	0.119776	0.119200
128	63	0.449376	0.450208
128	127	2.664448	2.664704
512	3	0.036864	0.036128
512	7	0.099872	0.093088
512	15	0.344064	0.343808
512	31	1.392640	1.393056
512	63	5.671136	5.669888
512	127	34.465759	34.467522
1024	3	0.103616	0.103392
1024	7	0.327648	0.327360
1024	15	1.320192	1.320704
1024	31	5.466080	5.465664
1024	63	22.419201	22.421633
1024	127	75.128700	61.306721
2048	3	0.176160	0.179904
2048	7	0.564416	0.566784
2048	15	2.322560	2.315360
2048	31	9.627648	9.632160
2048	63	38.678432	33.009056
2048	127	199.408249	198.885727
4096	3	0.583200	0.591072
4096	7	1.835104	1.836256
4096	15	7.627072	7.625280
4096	31	31.820160	31.821344
4096	63	130.675354	130.673050
4096	127	795.198669	795.185913
8192	3	2.318496	2.308160
8192	7	7.294400	7.300384
8192	15	30.438593	30.443359
8192	31	127.135551	127.143135
8192	63	523.571594	524.991150
8192	127	3194.325684	3194.307129
10000	3	3.439808	3.429184
10000	7	10.824160	10.831872
10000	15	45.470142	45.870174
10000	31	191.256577	190.060349
10000	63	781.314697	781.959045
10000	127	4749.800293	4738.020508

Discussion

Observations:

- For smaller matrices, the performance difference between the no-tiling and tiling approaches is negligible due to the low memory overhead.
- For larger matrices and kernels, the tiling approach significantly enhances performance by reusing shared memory more efficiently and minimizing global memory access.
- Larger kernel sizes approach the shared memory limit of the Tesla T4 GPU, emphasizing the importance of memory optimization strategies.

Limitations:

- Kernels larger than 127×127 may exceed the shared memory capacity, requiring additional optimization or the use of global memory.
- The results presented are specific to the Tesla T4 GPU architecture; further investigation is needed to assess scalability on other GPU models with different hardware characteristics.

Conclusion

This study highlights the significant performance improvements achieved by employing tiling for 2D convolution on GPUs. The tiling approach leverages shared memory efficiently, reducing the reliance on slower global memory and optimizing memory access patterns. While the performance difference is minimal for smaller matrices and kernels, the benefits of tiling become pronounced as matrix and kernel sizes increase, showcasing its scalability for larger computational workloads. These findings underscore the importance of shared memory optimization in GPU programming and demonstrate the potential of tiling to enhance computational efficiency in data-intensive applications.