GPU Computing Project

-Convolution Operation

1. 프로젝트 개요

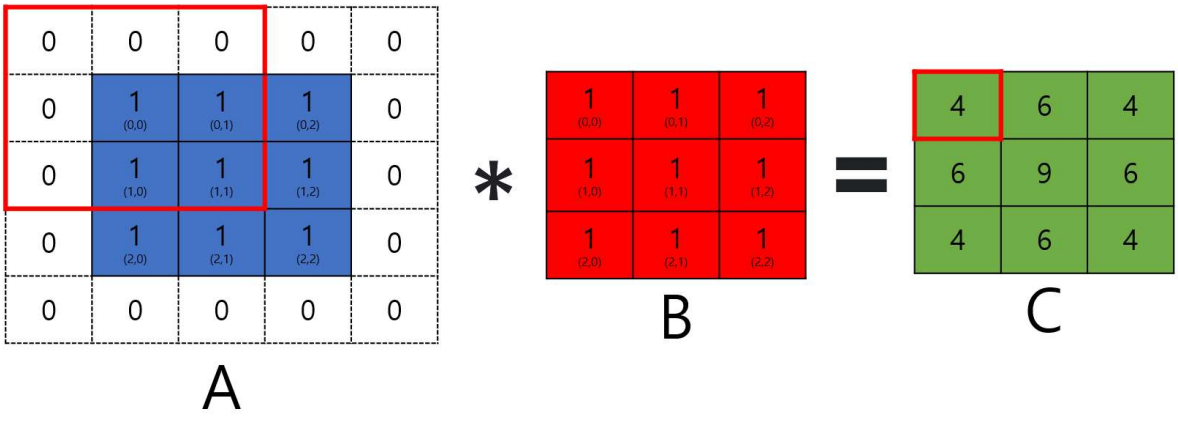
Convolution은 GPU 병렬프로그래밍 환경, 특히 Deep Learning이나 Artificial Intelligence(AI) 분야에서 자주 쓰이는 연산이다. 연산 자체는 간단하지만 굉장히 많은 데이터를 연산해야 하기 때문에 GPU를 이용한 병렬처리를 통해서 처리는 것이 효율적이다. 위 프로젝트에서는 CUDA를 사용해 Kernel을 구현해 정해진 Input Feature Map에 대해서 3개의 Weight Size에 대한 Convolution 연산을 수행하는 프로그램을 구현한다. 위 과제에서 Stride는 1로 가정한다.

Kernel의 성능은 수행하는 GPU의 특성, Kernel의 Memory Access Pattern, Thread 및 Block의 구성, 어떤 Memory를 사용하는지, Kernel 내에서 어떤 동작을 하는지 등 다양한 요소에 의해 달라지므로 구현하는 프로그래머의 GPU Computing에 대한 이해에 따라 성능이 결정된다. 위 프로젝트에서는 프로그램의 성능을 최대한 빠르게 하는 것에 목표를 둔다.

2. 배경지식

A. 2D Convolution

Deep Learning이나 AI에서의 Convolution이란 Elementwise Matrix Multiplication들의 합을 뜻한다.



$$C(i,j) = \sum_{M=-1}^{1} \sum_{N=-1}^{1} A(i+M, j+N) \times B(M+1, N+1)$$

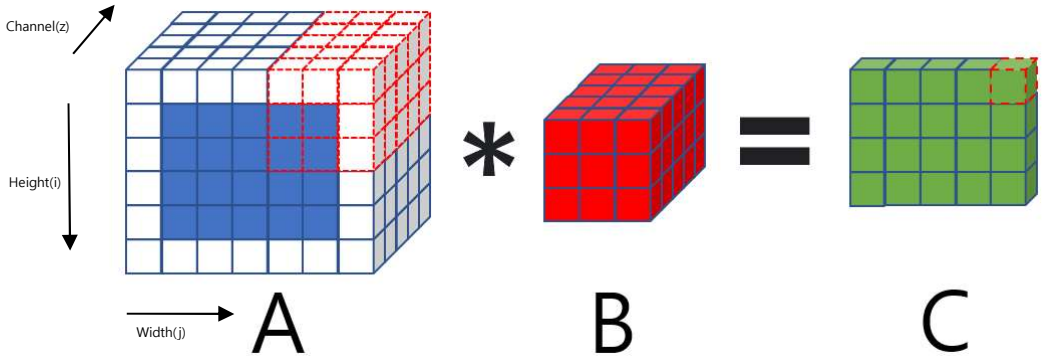
위 그림은 크기가 3x3인 Input Feature Map A와 Weight B의 2D Convolution 예시이다. A은 빨간 바탕은 Input Feature Map, A의 하얀색 바탕의 0는 연산을 위해 임의로 할당한 Zero padding이고 B는 Weight, C는 A와 B의 Convolution으로 인해 만들어지는 Output Feature Map이다.

Convolution연산은 Weight인 B가 Input Feature Map인 A를 움직이며 Weight내의 A의 Element와 B의 Element의 곱들의 합으로 Output Feature Map을 연산한다. 수식으로 예를 들면 아래와 같다.

$$C(0,0) = A(-1,-1) \times B(0,0) + A(-1,0) \times B(0,1) + A(-1,1) \times B(0,2) + \cdots + A(1,1) \times B(2,2)$$
$$C(1,1) = A(0,0) \times B(0,0) + A(0,1) \times B(0,1) + A(0,2) \times B(0,2) + \cdots + A(2,2) \times B(2,2)$$

B. **3D Convolution**

3D Convolution은 2D Convolution과 같지만 3D Weight를 사용하는 Convolution을 말한다.



위 그림과 같이 3D Input Feature Map인 A와 3D Weight인 B가 Convolution연산을 해 Output Feature Map C를 만든다.

$$C(i,j) = \sum_{V=0}^{4} \sum_{M=-1}^{1} \sum_{N=-1}^{1} A(z+V, i+M, j+N) \times B(V, M+1, N+1)$$

3. 프로젝트 설명

A. Input & Output

위 과제에서 Input Feature Map과 Weight는 구현되어 코드로 제공되며 아래 표와 같은 크기를 가진다.

	Input Feature Map	Weight	Output Feature Map
2D Convolution 3x3	224 x 224 x 300	3x3	224 x 244 x 300
3D Convolution 1x1	224 x 224 x 300	1x1x300x900	224 x 224 x 900
3D Convolution 3x3	224 x 224 x 300	3x3x300x900	224 x 224 x 900

이때 크기는 Width x Height x Channel(Depth) x Output Channel(필터의 수)를 뜻한다.

B. Implementation

위 Project에서 3개의 Kernel을 구현한다. 각 Kernel은 3x3, 1x1x300x900, 3x3x300x900의 Weight를 가지며 Input Feature Map과 Convolution연산을 해서 각각 224x224x300, 224x224x900, 224x224x900의 Output Feature Map을 만든다.

3가지 필터에 대한 연산을 각각 2D Convolution 3x3 , 3D Convolution 3x3, 3D Convolution 1x1이라고 명명한다.

2D Convolution 3x3은 3x3 Weight가 모든 Input Feature Map Element에 대해서 Convolution연산을 하며 같은 Channel의 Output Feature Map Element에 저장된다. 예를 들어 (1,1,3)를 중심으로 하는 Convolution의 결과는 (1,1,3)에 저장된다.

3D Convolution의 Output Feature Map의 Channel은 Weight의 Output Channel로 결정된다. 예를 들어 Input Feature Map Element와 Weight (x,x,x,5)의 Convolution 결과는 (x,x,5)에 저장된다.

그 외 사항들은 제공되는 코드의 주석을 참고한다.

4. 보고서

보고서는 Introduction, Background, Code explanation, Optimization, Conclusion으로 구성한다.

A. Introduction

프로젝트에 대한 전반적인 설명(5~10줄 이내)

B. Background

GPU의 구조 및 CUDA, CPU와 비교해 GPU에서의 Convolution 연산의 장점에 대해 자세하게 서술

C. Code explanation

구현한 Kernel에 대한 설명

D. Optimization

Kernel의 연산을 최적화하기 위해서 어떤 방법을 사용했는지, 코드에서 Block 및 Thread의 수를 정한 이유와 Memory를 어떻게 사용했는지 등 서술

E. Conclusion

프로젝트를 구현하며 어려웠던 것, 배운 것, 문제가 되었던 부분과 어떻게 해결했는지 등을 서술

5. 제출기한 및 배점

A. 배점

보고서 : 50%

코드 : 50%

B. 제출기한 및 양식

i. **제출기한 : 11월 16일 12:00시**

ii. **양식 :** 소스코드 파일 및 보고서(PDF파일)을 압축해 **Project_학번_이름**으로 제출
예시) Project_2015722030_박우혁