**System Programing**

Assignment #1 .Parallel CNN

**컴퓨터 공학과**

**201811242 김재우**

**Content**

1. Design ( 설계 ) & Implementation ( 구현 )

2. Function description ( 함수 설명 )

**1. Design ( 설계 ) & Implementation ( 구현 )**

이번 과제인 Parallel CNN을 구현 하기 위해 우선 공통적으로 Convolutional Layer 과 Max-pooling Layer를 짜기 위해서 가장 중심적인 알고리즘은 주어진 과제에서 filter의 값과 크기, Max-pooling할 경우 배열을 자르는 크기와 strides가 정해져 있고 ku\_func.o에 존재하는 makeMatrix라는 함수를 보면 이중 포인터의 형식으로 데이터를 만들어 주므로 matrix[i][j]의 형태로 데이터를 접근할 수 있게 됩니다. 그러므로 이를 이용하여 i, j 값, 즉 index값을 데이터로 하여 Message queue를 사용합니다. 데이터가 들어온 이 Message queue를 이용하여 index 데이터를 받고 각 layer를 계산시킨 뒤 데이터를 처리할 때 있어서 Message queue의 enqueue와 dequeue를 할 때 우선 순위를 정하기도 가편하다고 생각하였습니다.

로직에 대해 설명을 하면 처음 부모는 각 자식들에게 현재 계산 시킬 부분의 첫 index를 주어 Message queue1에 실어 보냅니다. 자식은 Message queue에서 받아온 id값과 value에 담긴 index를 이용하여 Convolution이나 Max-pooling을 하게 되는데 이때 계산된 값을 Message queue2에 넣어서 부모에게 보냅니다. 이때 Message queue의 크기가 제한되어 있어서 layer의 크기가 일정이상 커질 경우 Message queue에서 데이터를 못 받아오거나 프로세스가 너무 많이 생겨서 프로그램이 멈춰버리는 경우가 생긴다. 그러므로 부모에게 보낸 Message queue의 count가 4를 주기로 부모프로세스는 Message queue에 값을 가져옵니다. (이때 4를 주기로 설정해준 이유는 입력 값이 2i+2의 형태인데 이때 layer자체는 정사각행렬이므로 총 데이터의 개수는 (2i+2)^2=4(i+1)^2 가 되므로 4를 루프로 하는 구조로 쉽게 파악할 수 있었습니다.) 위와 같이 루프 구조를 이용하면 Message queue의 크기 제한이나 프로세스의 개수를 조절할 수 있어 큰 값을 사이즈로 입력 받을 수 있게 됩니다.

위 루프의 데이터 흐름을 보면 모든 데이터는 자식프로세스에게 Message queue를 통해 넘겨주게 되고 자식프로세스에서 Convolution, Max-pooling의 계산을 한 후 계산된 값을 다시 Message queue통해 부모에게 값이 전달되게 됩니다.

자세한 구현 코드는 아래에 각각 Convolution을 하는 함수와 Max-pooling을 하는 함수의 코드를 사진으로 넣어 놨습니다.

1.1 Convolutional Layer





2.1 Max-pooling



**2. Function description ( 함수 설명 )**

2.1 Convolution 과 Maxpooling을 시작하는 함수

|  |  |  |
| --- | --- | --- |
| Fuuction Name | Arguments | Description |
| maxPooling | Int\*\* matrix; | Max-pooling을 계산하기 위해 데이터가 들어 있는 행렬 |
| Int\*\* aftermat; | Max-pooling을 한 뒤 계산 값을 저장해서 보내주기 위한 행렬 |
| Int row; | 계산을 할 행렬의 가로(=세로) 값 |
| Int mqdes1; | 자식이 부모로부터 받을 데이터(계산할 인덱스)를 가지고 있는 message queue의 키 값 |
| Int mqdes2; | 자식이 부모에게 줘야할 데이터(계산 결과값)를 보낼 때 이용할 message queue의 키 값 |
| return void |  |

|  |  |  |
| --- | --- | --- |
| Fuuction Name | Arguments | Description |
| conv | Int\*\* matrix; | Convolution을 계산하기 위해 데이터가 들어 있는 행렬 |
| Int\*\* filter; | Convolution을 하기 위해서 필요한 filter 행렬 |
| Int\*\* aftermat; | Convolution을 한 뒤 계산 값을 저장해서 보내주기 위한 행렬 |
| Int row; | 계산을 할 행렬의 가로(=세로) 값 |
| Int mqdes1; | 자식이 부모로부터 받을 데이터(계산할 인덱스)를 가지고 있는 message queue의 키 값 |
| Int mqdes2; | 자식이 부모에게 줘야할 데이터(계산 결과값)를 보낼 때 이용할 message queue의 키 값 |
| return void |  |

2.2 Convolution과 Maxpooling중 index에서 계산 해주는 함수

|  |  |  |
| --- | --- | --- |
| Fuuction Name | Arguments | Description |
| convCal | Int\*\* matrix ; | Convolution을 계산하기 위해 데이터가 들어 있는 행렬 |
| Int\*\*filter ; | Convolution을 하기 위해서 필요한 filter 행렬 |
| Int i ; | Convolution을 계산을 시작할 행렬의 행 값 |
| Int j ; | Convolution을 계산을 시작할 행렬의 열 값 |
| return int | matrix[i][j]를 기준으로 filter만큼 Convolution을 한 계산 값 |

|  |  |  |
| --- | --- | --- |
| Fuuction Name | Arguments | Description |
| poolCal | Int\*\* matrix ; | Max-pooling을 계산하기 위해 데이터가 들어 있는 행렬 |
| Int i ; | Max-pooling을 계산을 시작할 행렬의 행 값 |
| Int j ; | Max-pooling을 계산을 시작할 행렬의 열 값 |
| return int | matrix[i][j]를 기준으로 정해진 Max-pooling의 region만큼 Max-pooling을 한 계산 값 |

2.3 동적할당 및 동적할당 정리 해주는 함수

|  |  |  |
| --- | --- | --- |
| Fuuction Name | Arguments | Description |
| makeAllocation | Int\*\* matrix ; | 동적 할당을 시켜줄 int\*\*타입의 matrix |
| Int row ; | 동적할당을 해줄 가로(=세로)의 값 |
| return int\*\* | 동적할당이 된 int\*\*타입의 값 |

|  |  |  |
| --- | --- | --- |
| Fuuction Name | Arguments | Description |
| freeAllocation | Int\*\* matrix ; | 동적 할당할 정리할 int\*\*타입의 matrix |
| Int row ; | 동적할당을 정리 할 가로(=세로)의 값 |
| return void |  |