Assignment6-cuda

컴퓨터전공
2013011491
안찬영

1. Implementation

-Max Pooling

```
int idx = row*input_size*filter_size+col*filter_size;
 float max=0;
   if(input_size-(col*filter_size)>=filter_size){
        if(idx<input_size*input_size){
             for(int i=0; i<filter_size;i++){
                  for(int j=0; j<filter_size; j++){
                       if(max<input[idx+j+i*input_size])
                            max=input[idx+j+i*input_size];
                  }
             }
        }
        output[col+row*(input_size/filter_size)]=max;
}
```

하나의 thread는 filter_size* filter_size matrix에서 max값을 뽑아내도록 구현하였습니다. 첫 번째 조건문에서 input_size가 filter_size의 multiple이 아닐 때 left_over 부분을 무시하기 위해 사용되었습니다. 그 다음은 filter_size로 나눠진 matrix에서 loop를 돌면서 최댓값을 output에 저장합니다.

-simple GEMM

```
if(col==(input_size-1)&&((col+1)%TILE_WIDTH!=0)){
                for(int i=1; (col+i)%TILE_WIDTH!=0; ++i){
                     if(row*input_size+p*TILE_WIDTH+tx+i>=(i*input_size)-1) continue;
                     s_a[ty][tx+i]=a[row*input_size+p*TILE_WIDTH+tx+i];
                }
           }
           if((p*TILE_WIDTH+tx)<(input_size)&&row<input_size){
                s_a[ty][tx]=a[row*input_size+p*TILE_WIDTH+tx];
           }else{
                s_a[ty][tx]=0.0;
           }
```

일부분만 설명하면 첫 조건문에서 input_size가 TILE_WIDTH의 multiple이 아닐 때, 행렬 마지막 column index를 찾아냅니다. 이 경우에 block에서 tx가 filter_size보다 작기 때문에 s_a[ty][tx]의 배열에 없는 tx의 대해서 값을 불러오지 않기 때문에, 이 때에만 연산이 추가

로 행해지게 됩니다. 나머지 부분에 대해서는 tile 사이즈 만큼 공유메모리에 값을 불러와 block 단위로 계산하게 됩니다.

2. performance