AM148 Final

Winson Chen

June 2022

1 Background

Since I have huge interest in Convolution Neural Network and had a project on wanting build a library from scratch, and want to know the architecture of the Convolution Neural Network by building it on my own. I want to compare the timing between my GPU implementation and CPU implementation in modern library between what I built before. By building from scratch I am able to learn about how to write optimization algorithm in GPU and the Neural Network architecture. I have overestimate that I could also work on the back propagation part of the architecture, so I only implemented Convolution, Maxpooling, and Dense Layers.

In my program, I am using 3 GPU algorithms to parallelize the process:

Algorithm 1: Parallel Convolution with padding 0 around ($O(n^2)$ n size of filter)

```
Data: $input, filter, row, col, i, j, N, sum$
    $\triangleright$ $row, col$ are thread location grid,$N$ is number of threads;
Result: $C$
Using $N$ threads:
Thread $t$ reads $input, filter$;
$sum = 0$;
for $i = -1 \rightarrow filter.y - 1$ do
    for $j = -1 \rightarrow filter.x - 1$ do
        $sum += input[row - i + 1][col - j + 1] * filter[i + 1][j + 1]$ ;
    end
end
Thread $t$ exports $C[row][col]$;
```

Algorithm 2: ParalMaxPooling ($O(log_2(N))$)

```
Data: $input, poolsize, row, col, N, sid, sdata, max, block$
        ▷ $row, col$ are thread location grid,$N$ is number of threads;
Result: $C$
Using $N$ threads:
Using $M$ blocks: Thread $t$ reads $input, poolsize$;
each block loads $poolsize$ block from $input$ to $sdata$;
for $s = poolsize * poolsize/2$ to $s > 0$ $s/ = 2$ do
    if $sid < s$ then
        if $sdata[sid] > sdata[sid + s]$ then
            $sdata[sid] = sdata[sid]$;
        else
            $sdata[sid] = sdata[sid + s]$;
        end
    end
end
if $sid == 0$ then
    $max = sdata[0]$ if $max < sdata[poolsize * poolsize - 1]$ then
        $max = sdata[poolsize * poolsize - 1]$;
    end
    $C[block.x][block.y] = max$;
end
```

Algorithm 3: Parallel matmul $O(O(N))$

```
Data: $t, i, j, A, B, N$
                        ▷ $N$ is number of threads;
Result: $C$
Using $N$ threads:
Thread $t$ reads $A[t], x$;
for $Thread\ t\ in\ Total\ thread\ N$ do
    $Sum = 0$;
    for $j\ in\ N$ do
        $Sum += A[t][j] * B[j][t]$;
    end
    $C[t][i] = Sum$;
end
Thread $t$ exports $C[t]$;
```

2 Plans

My plan of attack is to make the Convolution Layer and all the other layers in the CNN to bdardare able to parallelize and able to run on GPU. Parallelizing these tasks:

- Convolution Layer: splitting up each filter into blocks to run the convolution on the image or divide up more tasks for GPU to compute.
- MaxPooling Layer: extract the features and reduce the size of the output from convolution layer which can compute subsections of the layer and gather it to form a Maxpooling layer.
- Dense Layer: since is fully connected Neural Network, so we can use parallelized matrix multiplication to optimize the operation of feedforward.

3 Result

This result is showing that I used two Conventional, one Maxpooling, and one Dense layer to do a demo of how each layer serve as:

You can run this following line and get the output:

```
$ ./test.sh CNN
```

You can also run the different layer by itself that you can just uncomment the

```
int main() { ... }
```

and run:

```
$ ./test.sh <name of the layer> (conv, maxpool, dense)
```

```
filter1
Shape: 3x3
0 1 0
0 1 0
0 1 0

filter2
Shape: 3x3
1 0 0
0 1 0
0 0 1

img
Shape: 6x6
1 1 1 1 1 1
1 1 1 1 1 1
```

```
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1

First Conv2D
dimBlock: 32x32
dimGrid: 1x1
GPU convolution Time = 0.033824ms
Shape: 6x6
2 2 2 2 2 2
3 3 3 3 3 3
3 3 3 3 3 3
3 3 3 3 3 3
3 3 3 3 3 3
2 2 2 2 2 2

Second Conv2D
dimBlock: 32x32
dimGrid: 1x1
GPU convolution Time = 0.009184ms
Shape: 6x6
5 5 5 5 5 2
6 8 8 8 8 5
6 9 9 9 9 6
6 9 9 9 9 6
5 8 8 8 8 6
2 5 5 5 5 5


MaxPooling2D
dimBlock: 2x2
dimGrid: 3x3
GPU MaxPooling Time = 0.009216ms
Shape: 3x3
8 8 8
9 9 9
8 8 8

Dense Layer
Naive GPU MatMul Time = 0.009248ms
Shape: 3x4
16.2369 9.16723 12.7658 17.5643
18.2665 10.3131 14.3615 19.7599
16.2369 9.16723 12.7658 17.5643
```

4 Conclusions

I could further optimize my algorithm for convolution and matmul to shared memory and able to do tiling for the operation. I think that finding better splits for maxpooling to compare the maximum number can also be find and apply. GPUs had benefit my project by speeding up the process and allowing me to know how to utilize the hardware to its potential in order to make the use of it. I have also learned that it is really challenging to create GPU algorithm for some task that would be easy for serial implementation.

5 Source

Convolutional Neural Networks, Explained.

A Convolution Neural Network (CNN) From Scratch.

CNN from scratch(numpy).

How do I allocate memory and copy 2D arrays between CPU / GPU in CUDA without flattening them?.