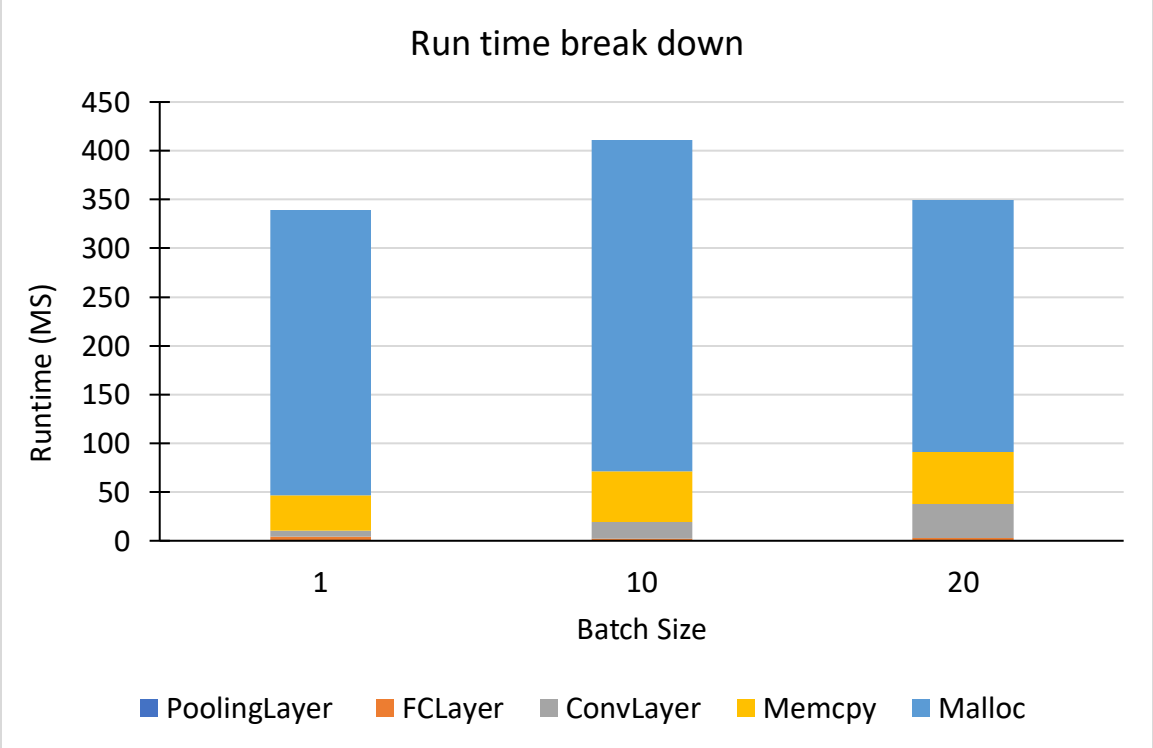
CUDA Programming Assignment Part 4 Report

In the last CUDA programming assignment, I implemented AlexNet in CUDA. In the previous assignments, I implemented each layer of the AlexNet. In the last part, I combined the layers to build an end-to-end AlexNet inference only network.

To build the network, I create APIs for Pooling, conv and FC layers. Then, I hard code the size of each layer in the header file. Then, I created the

To verify the neural network, I tried to verify the output layer by layer compared against the CPU. I also measured the runtime and found the sum of runtime of all layers are similar to the kernel runtime of the AlexNet. I checked the memory with cude-memory-check and find no error.

This figure shows the breakdown of the performance. It is clear that the CUDA Malloc API consumes most of the runtime. The memory copy also takes a long time. To enhance the performance, I tried to combine the layers together to reduce the kernel launch time. However, because most of the time is on Malloc, the optimizations in the kernel really does not make a difference.



To optimize the network, I tried to malloc the CUDA memory and keep it in the GPU for a few batches. With batch size = 8, I tried to run several batches to see the performance. As the number of batches grows, the kernel runtime (Pooling + FC + Conv) starts to domination the runtime. The overhead of malloc is amortized with multiply batches.

I think the performance can be further optimized by finding the best block size and number of blocks for each layer. However, I did not want to spend that amount of time to enhance the performance.