WiDS Week 2

Task 1: Implement Basic CUDA Kernels

Three basic CUDA kernels vector addition, elementwise multiply-and-scale, and ReLU were implemented. Each kernel allocates GPU memory with cudaMalloc, transfers data using cudaMemcpy, launches a one-dimensional grid with explicit bounds checking, and copies results back to the host. Correctness was verified by comparing GPU outputs against a CPU reference implementation, and all programs compiled cleanly with nvcc without warnings.

Task 2: Grid and Block Configuration Exploration

gridSize = (n + blockSize - 1) / blockSize

Vector size n = 1000

blockDim.x = 32	gridDim.x = 32	threads launched = 1024
blockDim.x = 128	gridDim.x = 8	threads launched = 1024
blockDim.x = 256	gridDim.x = 4	threads launched = 1024
blockDim.x = 512	gridDim.x = 2	threads launched = 1024

Vector size n = 100000

blockDim.x = 32	gridDim.x = 3125	threads launched = 100000
blockDim.x = 128	gridDim.x = 782	threads launched = 100096
blockDim.x = 256	gridDim.x = 391	threads launched = 100096
blockDim.x = 512	gridDim.x = 196	threads launched = 100352

Vector size n = 10000000

blockDim.x = 32	gridDim.x = 312500	threads launched = 10000000
blockDim.x = 128	gridDim.x = 78125	threads launched = 10000000
blockDim.x = 256	gridDim.x = 39063	threads launched = 10000128
blockDim.x = 512	gridDim.x = 19532	threads launched = 10000384

2.1 What happens if the total number of threads exceeds n?

When the total threads launched exceeds n, we have threads which do not do any computation whatsoever. They just exit without performing any memory access.

2.2 Why is bounds checking required inside the kernel?

Bounds checking is required to prevent threads with thread indices greater than n from accessing memory beyond the original array. Since CUDA launches threads in blocks, it is possible for the total number of threads to exceed the size of the array making it necessary to check the bounds for correctness.

2.3 Which configurations failed or behaved unexpectedly, if any?

Larger block sizes caused more threads to be launched than what was required. Smaller block sizes actually ended up launching the exact number of threads. All configurations worked though.

2.4 Based on Task 2, when might you choose a smaller block size vs. a larger one? What trade-offs do you anticipate?

Smaller block size may be preferred when we need to minimise the number of excess threads. Larger block sizes can be useful for better resource utilisation.