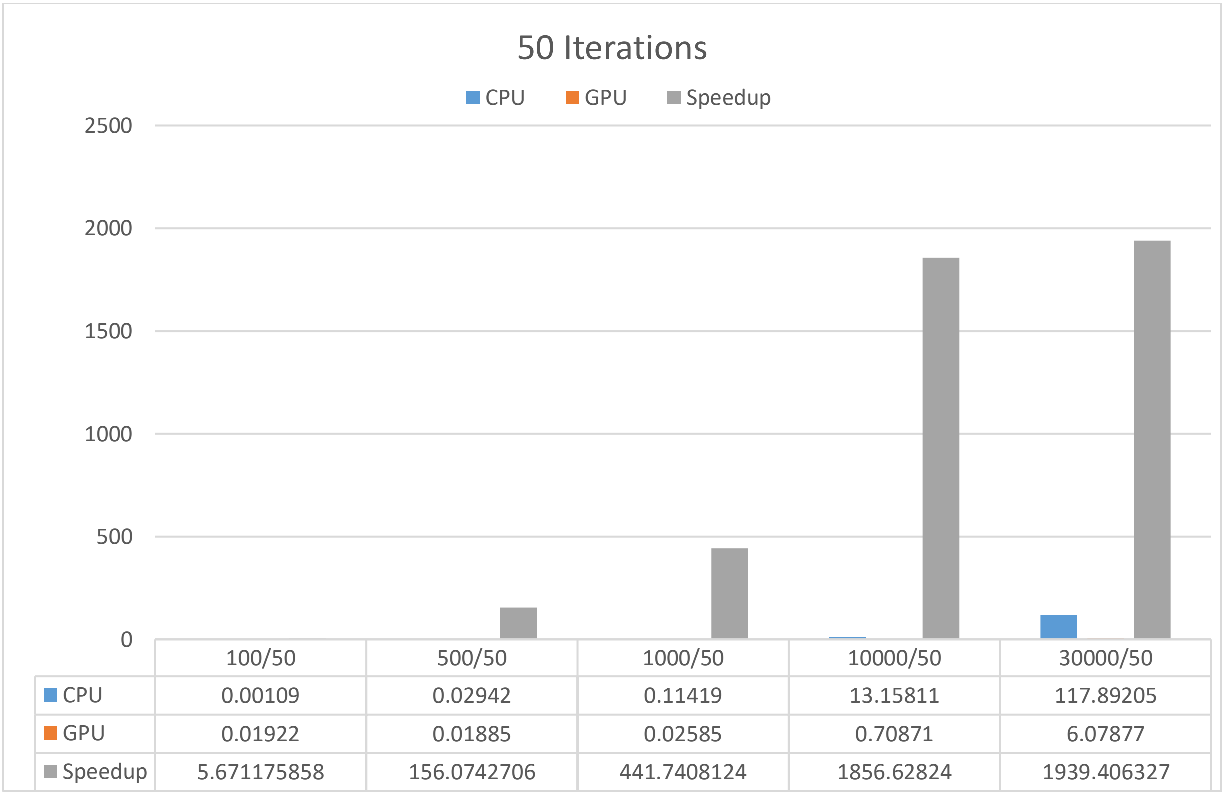
CS398 Assignment 1 report

Analysis of CPU vs GPU usage

The below figure shows a bar graph plotting time taken for 50 iterations and speedup from CPU to GPU. The data table is included as certain values are unable to be seen due to the difference in values. The speedup section is also included to be discussed later.



	100/50	500/50	1000/50	10000/50	30000/50
■ CPU	0.00109	0.02942	0.11419	13.15811	117.89205
■ GPU	0.01922	0.01885	0.02585	0.70871	6.07877
■ Speedup	5.671175858	156.0742706	441.7408124	1856.62824	1939.406327

Figure 1. Bar graph plotting time taken for 50 iterations and speedup from CPU to GPU.

As seen from the above figure, the GPU computations starts having a positive speed up of 56% after 500 elements with 50 iterations. This hence shows that the GPU usage is more beneficial at 500 elements as that is the first number of elements which the performance increases.

The lowest speedup as seen from the figure is at 100 elements. This is due to the bottleneck caused by GPU processors instruction latency. As seen from the other iterations, the change in n elements does not affect the GPU's speed as much as it does for CPU. Therefore, we can conclude that since the GPU has a minimum latency, the latency is significantly more than the time taken to calculate 100 elements which causes it to have a negative speed up of 95%.

The speed up is the highest at 30,000. However, in this scenario, the latency is not the same as 10,000 elements. This is due to the GPU hardware limitation and the algorithm must split the data to batch instead of having a full contiguous memory. However, we can still see that even though we split the data to do batching, the speed up is still very high. This is due to the latency being significantly lesser than the calculation of the huge number of elements in the array, which takes a lot of time in the CPU compared to the GPU.

500 iterations using 1000, 10000 and 30000 elements

	CPU	GPU	CPU Percentage	GPU percentage	Speedup
1,000	1.78574	0.24899	87.76%	12.24%	717.19%
10,000	138.9565	6.45735	95.56%	4.44%	2151.91%
30,000	1204.62	56.07944	95.55%	4.45%	2148.06%

Table 1. Table of elements for 500 iterations

The effects of increasing the number of iterations does not scale as according to how we would assume. For example, iterating twice should speed it up by twice. It does not speed it up but in fact hovers around the same percentage as the previous 50 iterations counterpart. This is due to the inbuilt latency of sending instructions to the GPU from the CPU which is done per iteration. Therefore, more iterations means more latency, resulting in almost the same amount of speed up in comparison to 50 iterations.