EE 5545 Assignment 3

Ethan Glaser (eg492)

I. Problem 1

	Colab	Leaderboard
Original non-optimized implementation	10.12ms	152.8s
Optimized implementation	0.2488ms	0.1343s
Speedup	40.68	1137.75

Optimizations:

- Padding – this reduces the operations dependence on if statements, by initially padding to yield a final result that matches the shape of the input a repeated process can then be vectorized and parallelized as well as operated on without satisfying conditions originally described
- Parallel – division of operations across multiple threads can speed up computation, with similar operations happening simultaneously
- Vectorize – uniform memory access patterns with 1D convolution can be drastically sped up by vectorization.

II. Problem 2

Runtime and speed-up according to leaderboard repository:

- Original non-optimized implementation – unknown – errored out when no bind for GPU
- Optimized implementation – 3.00s
- Speedup just from adding threading – 4.436

Optimizations:

- Padding – this reduces the operations dependence on if statements, by initially padding to yield a final result that matches the shape of the input a repeated process can then be vectorized and parallelized as well as operated on without satisfying conditions originally described
- Blocking – by breaking down large convolutions into blocks, operations can be done more efficiently in the GPU in parallel and then brought back together
- Threading – splits up work on blocks by assigning operations to a specific thread, increasing overall parallelism that again is ideal on a GPU

III. Problem 4

Runtime and speed-up according to leaderboard repository:

	Colab	Leaderboard
Original non-optimized implementation	519.24ms	6.97s
Optimized implementation	10.175ms	2.776s
Speedup	51.03	2.51

Optimizations:

- Blocking – by breaking down large convolutions into blocks, operations can be done more efficiently in the GPU in parallel and then brought back together
- Threading – splits up work on blocks by assigning operations to a specific thread, increasing overall parallelism that again is ideal on a GPU
- Parallel – division of operations across multiple threads can speed up computation, with similar operations happening simultaneously
- Vectorize – uniform memory access patterns with 1D convolution can be drastically sped up by vectorization.
    - Note that parallel and vectorize did not yield much benefit since both are already taken into account when using a GPU

IV. Problem 5

Runtime and speed-up according to leaderboard repository:

- Optimized implementation (Colab): 0.163ms
- Optimized implementation (leaderboard): 3.797s
    - Note that there were no baseline implementations to compare to so speedup is not included

Optimizations

- Padding – this reduces the operations dependence on if statements, by initially padding to yield a final result that matches the shape of the input a repeated process can then be vectorized and parallelized as well as operated on without satisfying conditions originally described
- Blocking – by breaking down large convolutions into blocks, operations can be done more efficiently in the GPU in parallel and then brought back together
- Threading – splits up work on blocks by assigning operations to a specific thread, increasing overall parallelism that again is ideal on a GPU