**CS 521 ML & Compilers Spring 2025 – MP1**

**Student Name: Mingjun Wu**

**NetID: mw128**

**Part 1 CPU**

1.2) **Ablation Study:**

**O1-O3**

**A computer screen with white text

AI-generated content may be incorrect.**

**O4**

A computer screen with white text

AI-generated content may be incorrect.

A graph with blue and orange bars

AI-generated content may be incorrect.

Figure 1: Speedups of the Optimized Code

The results on the speedup graph show that code optimization improves performance overall, especially for larger matrix sizes. O0, the naïve implementation, serves as a baseline. O1 improves cache locality by reordering the loops. Tiling is introduced in O2 to improve cache efficiency by working on smaller submatrices. OpenMP parallelization and SIMD vectorization are used in O3 which allows parallel computation. Finally, compiler optimizations are used on O4 to maximize instruction-level parallelism. To be noticed, O4 is slower than the naïve implementation O0 for size = 100, which suggests that complier optimization might introduce overhead does not fully utilize the optimization which results in worse performance for smaller matrices. Similarly, O2 and O3 are slower than O1 for size = 100 and size = 1000 because tiling introduces more loop structures and parallelization using OpenMP can add thread management overhead.

1.3) **Scaling Study:**

***Insight:*** one paragraph describing the results above

**Part 2 GPU**

2.2) **Ablation Study:**

***Insight:*** one paragraph describing the results above

2.3) **Scaling Study:**

***Insight:*** one paragraph describing the results above