Yung Chi Liu
A20364639

# 1. Design.

1. Use pointer with size N*N ( N is high and width of array)

```
A = (float*)malloc(sizeof(float) * N * N);
B = (float*)malloc(sizeof(float) * N * N);
```

2. Use clock_t to time

```
clock_t time = matrix_normalization(A, B, N);
```

3. In function matrix_normal_CUDA

```
__global__ void matrix_normal_CUDA(float *A, float *B, float* means,
                                   float* sigma, int lineOfRow, int n )
```

float *A → input array with pointer.

float *B → result array pointer.

float *means → 1 dimension array(pointer) store each column's means.

float *sigma → 1 dimension array(pointer) store each column's calculation of sigma += powf(A[row][col] - means, 2.0);.

Int lineOfRow → Use in locate the position in pointer, in this case which is N (size of each row).

And last put result of

B[row][col] = (A[row][col] – mean) / standard_deviation

In *B.

In calculation, A[row][column] will translate to pointer with row = location/ lineOfRow and column = location% lineOfRow.

## 2. Testing and Result

| (ms) | Original | CUDA on Javis | CUDA on local NB |
|---|---|---|---|
| | | | Geforce GTX 950M |
| test1 | 7717.86 | 360 | 1146 |
| test2 | 7833.79 | 390 | 1091 |
| test3 | 7992.24 | 390 | 1134 |
| test4 | 7676.42 | 350 | 1126 |
| test5 | 7594.64 | 360 | 575 |
| Average | 7762.99 | 370 | 1014.4 |
| | | Speed Up | Speed Up |
| test1 | | 21.4385 | 6.73460733 |
| test2 | | 20.08664103 | 7.180375802 |
| test3 | | 20.49292308 | 7.047830688 |
| test4 | | 21.93262857 | 6.817424512 |
| test5 | | 21.09622222 | 13.20806957 |
| Average | | 20.98105405 | 7.652789826 |