Matrix Multiplication Project

- **Project Idea**

  Utilize CUDA to make an application that can multiply matrices

- **Explain Matrix Multiplication**

$$A_{2x3} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \qquad B_{3x2} = \begin{bmatrix} g & h \\ i & j \\ k & l \end{bmatrix}$$

$C = A \times B$

1) FIRST Check if they can compatible

A is 2x3 B is 3x2

Because the amount of columns in A match the amount of rows in B they can be multiplied

2) Determine C, the resultants, size

A is 2x3 B is 3x2

Size of C is 2x2

3) Set Up and Solve

$$C = \begin{bmatrix} ag+bi+ck & ah+bj+cl \\ dg+ei+fk & dh+ej+fl \end{bmatrix}$$

- **Project Purpose**

  To gain an understanding of CUDA and GPU parallelism

- **Tools**
  - Google Colab
  - Nvidia GPU
  - Cuda toolkit
  - Numpy
  - Cupy
  - PyTorch
  - Matplotlib
  - C++

•PSEUDOCODE

1. Import libraries
2. Define the GPU kernel
   ↳ code determining matrix compatibility and result dimensions
3. CPU code
   ↳ main()
      ↳ define matrix dimensions
      ↳ allocate memory on CPU
      ↳ Initialize matrices with data
      ↳ Allocate memory on GPU
      ↳ Use 'cudaMemcpy' to copy CPU data to GPU
      ↳ threads = , blocks =
      ↳ Launch Kernel
      ↳ Use 'cudaMemcpy' to copy GPU data to CPU
      ↳ Print
      ↳ Free CPU and GPU (because limited memory/good practice)

•Reflection

This project definitely had a bigger learning compared to the deep learning projects. I wish I had made the pseudocode a lot more in depth because I had to look up a lot of things. Nonetheless, this was my first time ever with CUDA and I feel that I now have a better understanding of how it all works. To summarize it you have to define the data in CPU, allocate it for both CPU and GPU, copy the data CPU → GPU. Threads are used to compute small parts of the data (the block contains the threads). This is a very minimal outline but it explains the idea of how CUDA and GPU parallelism works.