Plain vs Vectorized Matrix-Matrix Multiplication in python

Cole Chiodo

1. Introduction

The purpose of this experiment is to fully visualize and understand why using the tensors in the pytorch library for matrix multiplication is much more efficient than doing the same thing inside of plain python.

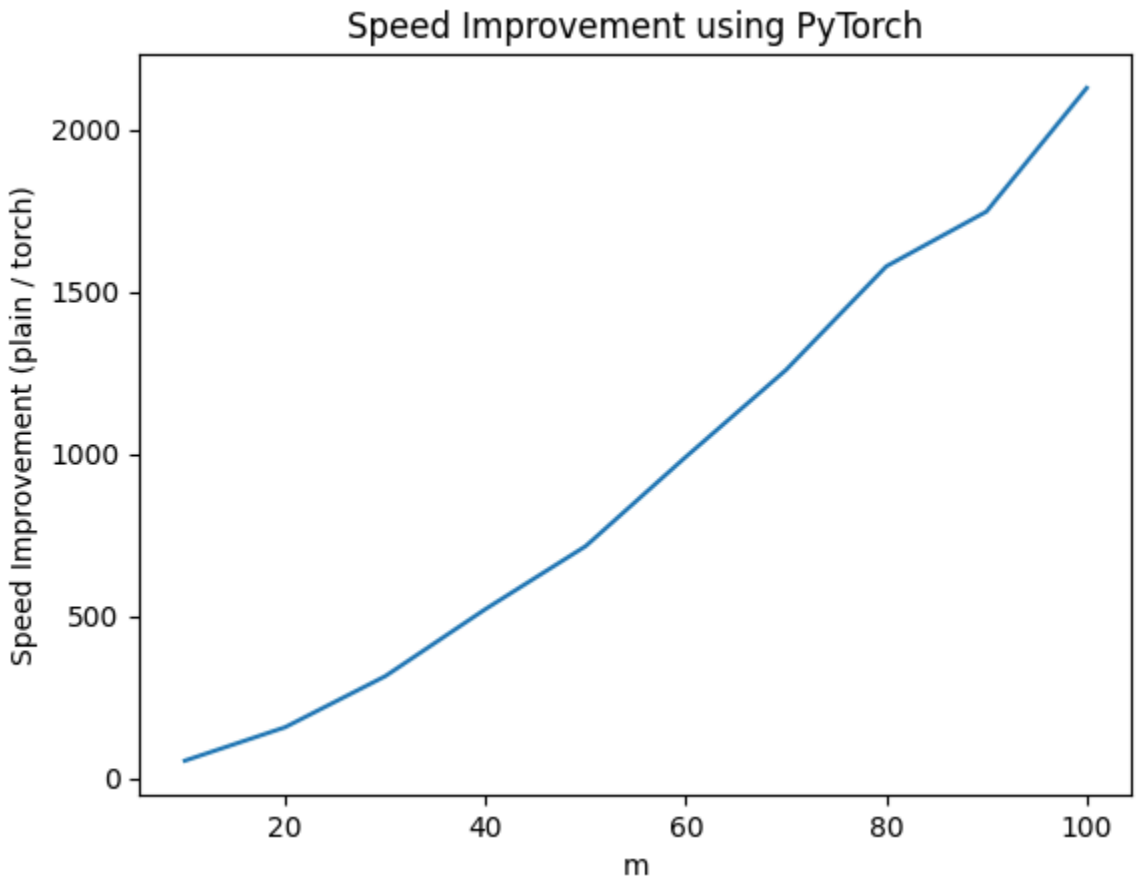
2. Approach

First is building the function to perform the plain matrix-matrix multiplication in python. This is a very simple function. It goes through each column and row of the two matrices, multiplying them in accordance to the dot product.

Next was to run the tests. I start by making a loop go through 10 times. Each i loop, it creates 2 matrices, the first being the X matrix with size of [i * 10, 110], the other being the W matrix with size of [90, i * 10]. Each loop then I input the two matrices into the function I made, making use of the %timeit function to get the data I want. Then I transform the data into tensors and use the tensor.matmul function, also recording the times.

After the loops, I print the results on a graph.

3. Results



As shown by the graph above, as the size of the data grows, the speed improvement from using tensors vs plain python is astronomical. Even when m is only 10, the speed improvement is nearly 100 times. Growing that data, just a little, to 100 makes that improvement over 2000 times greater.

4. Conclusion

In conclusion, this experiment clearly showed the speed improvements of using tensors in the pytorch library vs plain python. The results are due to how the code is ran. In plain python, the math is done through the CPU. Although powerful, the CPU is not quite optimized for large scale matrix dot product multiplication.

However, pytorch's tensors run all the math inside of the GPU. The GPU was created to do matrix multiplication in the most efficient way possible in order to render computer

graphics. We can use this fact to utilize the GPU to speed up the time it takes to train a model using deep learning.