Comparison of NumPy vs Python Implementaion of Strassen's Algorithm for Embedded Devices

Vishakha Dixit
ECE Department - 801265288
University of North Carolina at Charlotte
Charlotte, NC
vdixit2@uncc.edu

***Abstract*—This document explores simple experiments that contrast two Python Implementaion of matrix-matrix multiply. It also compares their performance metrics to help understand which approach is most suitable for small Embedded Devices like Raspberry Pi.**

***Index Terms*—NumPy, Strassen, Embedded, RaspberryPi**

I. Introduction

Python first emerged in the 1990s and has steadily gained popularity among software developers since then. Initially, Python became popular amongst embedded developers as a scripting language to test electronic devices. Slowly it has been moving further down the development stack. The Python programming language provides a rich set of high-level data structures: lists for enumerating a collection of objects, dictionaries to build hash tables, etc. However, these structures are not ideally suited to high-performance numerical computation. The NumPy package in python provides better ways of handling data for Mathematical Operations. The questions here is, does NumPy also enable efficient use of resources for Embedded Devices such as Raspberry Pi, where we have limited RAM, Flash memory, and CPU power consumption restraints?

II. Background

The Raspberry Pi 3 module is used for the purpose of this research. It is a low-cost single board computer that enables people of all ages to explore computing, and to learn about how to program in languages such as Python. Raspberry pi can also be used for development of complex applications such as Motion detection using webcam. Motion detection requires processing of every frame captured by the Raspberry pi camera module. Processing of frames can be CPU and Memory intensive, also power consumption for such application can be very high. This experiment explores different implementations using python and NumPy package.

III. Description of Experimental Infrastructure

Image processing involves extraction, filtering, enhancement etc. of images using mathematical operations. Every digital image has a corresponding matrix of color and color intensities. Various mathematical operations are performed on these matrices for enhancing the corresponding image.

This experiment uses Python programming language and two Python implementations of matrix-matrix multiply. Both these methods solve same problem of matrix multiplication. This experiment compares Python Implementation of Strassen's Algorithm with NumPy library's matrix multiplication function. The results are measured by comparing Execution time, CPU Utilization, and RAM Utilization.

The execution time parameter is being measured to quantify for how long the system needs to be in high power state to process the data. CPU Utilization measures how efficiently the process utilizes available CPU. Memory Utilization measures how much RAM is used by the process.

Python's psutil library is used to measure CPU and RAM Utilization, whereas time library is used to compute the execution time of the process.

A. Scope

This experiment uses following three different matrix dimensions:

- 64x64: represents low resolution image
- 256x256: represents an image closer to 360p resolution
- 1024x1024: represents high resolution image closer to 1080p

The dimensions taken here are in powers of 4 to create square matrix because Strassen's Algorithm uses recursive approach for matrix multiplication where in each recursive step it divides the matrix into 4 sub matrices of dimensions n/2 x n/2.

IV. Results And Analysis

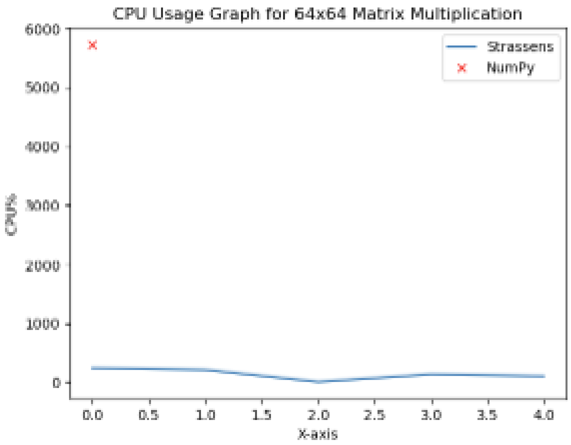
Following results were obtained after running both the Algorithms for all three dimensions mentioned in the Scope:

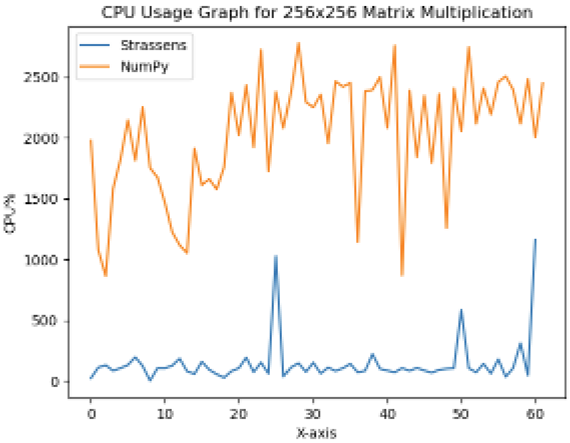
Algorithm	**64x64**	**256x256**	**1024x1024**
Strassen's	22884352	26177536	63369216
NumPy's Matmul	22794240	24301568	48500736

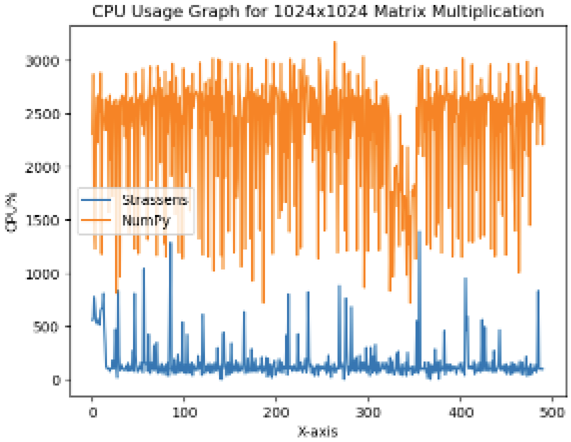
TABLE I
Memory Usage in Bytes

Algorithm	64x64	256x256	1024x1024
Strassen's	3261.342	155599.74	timed-out
NumPy's Matmul	5.797	1794.550	222168.144

TABLE II
EXECUTION TIME IN MS







Strassen's Algorithm got progressively slower with increasing size of matrix. For matrix of size 1024x1024, Strassen's Algorithm couldn't complete it's execution and timed out after running for 10 minutes.

For NumPy implementation Cpu runs at full computational power but for less amount of time. Stranssen's algorithm uses more memory and is efficient while using the CPU.

V. Conclusion

Numpy utilizes CPU resources more efficiently, hence the execution time required by NumPy is very low compared to Strassen's Algorithm. Therefore, the devices can be kept in low power sleep mode for longer period of time. Also, memory utilization in NumPy is less compared to regular python implementaion. Hence it is worth to use NumPy in an Embedded device, even though it might consume more Flash memory.

References

[1] **[NumPy]** Stefan van der Walt; S. Chris Colbert; Gael Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science and Engineering, Volume: 13, Issue: 2, March-April 2011, doi:10.1109/MCSE.2011.37.

[2] **[IPython]** Fernando Perez, Brian E. Granger. IPython: A System for Interactive Scientific Computing, Computing in Science and Engineering, vol. 9, no. 3, pp. 21-29, May/June 2007, doi:10.1109/MCSE.2007.53.