Designing Image Processing Library with accelerated matrix multiplication

Kaustubh Prakash & Abhay Saxena

Indian Institute of Technology, New Delhi

{*mt1160647,mt6160648*}*@iitd.ac.in*

17th Feb, 2019

Abstract

Large scale floating point matrix multiplication is used in many fundamental scientific and engineering applications. This report describes the building of an `Image Processing Library` using accelerated matrix multiplication. We have built `LeNet` [1] which is a tested Convolutional neural network and will test our data on the `MNIST` database. Different matrix multiplication techniques such as simple multiplication, using libraries like `Intel mkl` [2] and `openBLAS` [3] and a `pthread` implementation have been used in the CNN and their comparison has been tested.

1 Introduction

Image processing has advanced a lot especially in the field of machine learning where Convolutional Neural Networks have been proved successful to predict human-like learning patterns. The LeNet architecture was one of the first CNN's which predicted handwritten digits with a very low error rate leading to its development. However, all the image processing that is done requires multiplying large matrices which decreases the running time of most algorithms (like CNN's). Hence, we have tried to use different techniques of matrix multiplication and compared them to get an improved framework for image processing.

2 Linear Algebra Libraries

2.1 MKL

The Intel MKL [2] is one of the most highly regarded library to optimize code and to accelerate math processing routines. It is designed to be compatible to one's choice of compiler, language and operating system. It has the `mkl::cblas_sgemv` function which can multiply two matrices very quickly.

2.2 OpenBLAS

OpenBLAS [3] is an open source math implementation of the BLAS API with many hand-crafted optimizations for specific processor types. It also has a `mkl::cblas_sgemv` function to accerlerate matrix multiplication

3 Pthread Implementation

The simple matrix multiplication algorithm only utilizes one core of our CPU. Using concepts of multithreading, we can create several threads to evaluate different parts of the matrix multiplication. This leads to an overall decrease in running time. `Pthreads` are threads in C++ in which each thread has been equally allocated to do matrix multiplication (as shown in Figure 1). This operation has been implemented inside the convolution function.

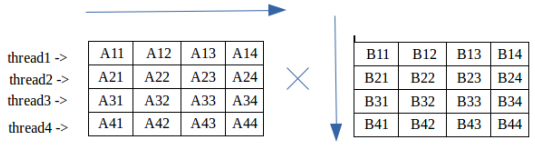
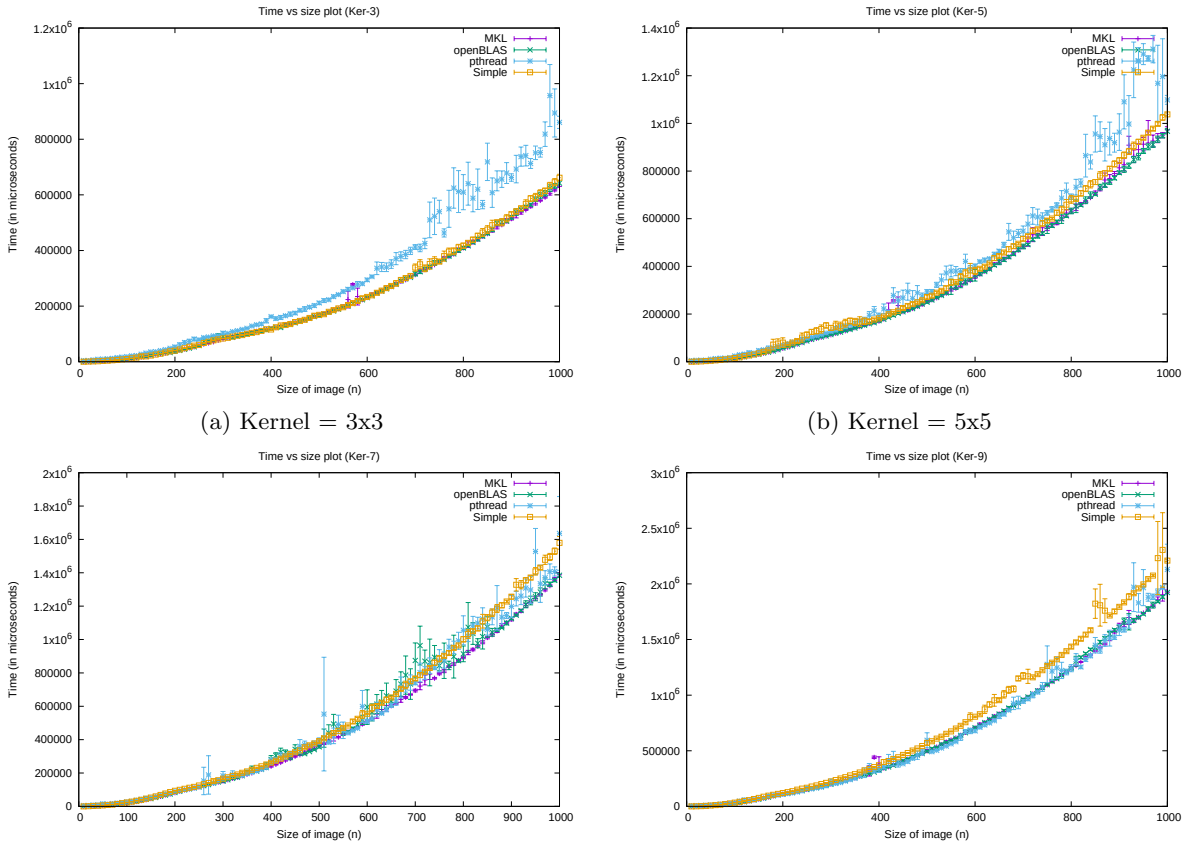


Figure 1: Matrix multiplication using Pthreads [4]

4 Comparison Tests

- For comparing convolution using different kinds of matrix multiplication algorithms, a simple code was made in `imgOp.cpp` called `conv_mult_pad`. This function takes in a `mult` input. The following values of `mult` calls the following multiplication algortihms.
  1. Simple Multiplication
  2. MKL multiplication
  3. OpenBLAS multiplication
  4. Pthread implementation of multiplication
- Then 10 matrices each with size from the range of 10 to 1000 were created with steps of 5 using `matgen.cpp`
- Using `evaluator.cpp`, 4 different kernel sizes *(3x3, 5x5, 7x7, 9x9)* were run on all the matrices and the running time was observed. The `mean` & `standard deviation` of the running times of the 10 matrices were calculated and their plots were obtained (in figure 2)



(a) Kernel = 3x3

(b) Kernel = 5x5

(c) Kernel = 7x7

(d) Kernel = 9x9

Figure 2: Time vs Size Plots

- These results were run on an Intel i7 6th gen processor with 8 cores, and hence 8 threads were used.

5 Observations

- From the plots (figure 2), we can observe that the Intel MKL [2] and OpenBLAS [3] methods of matrix multiplication were the fastest for all the kernels with minimal error. All the plot seem to have $O(n^3)$ nature. *Note*: The optimised libraries are faster than $O(n^3)$ algorithms.
- We can also notice that the pthread implementation reaches close to the faster implementations (calling the linear algebra libraries) when the kernel sizes are bigger. However, initially it is even slower than simple multiplication for smaller kernels due to large overheads of creating pthreads.
- However, it's clear that all these implementations are much faster than simple matrix multiplication for larger matrices. Only for fig 2 (a), speed of simple multiplication almost matches that of the optimised libraries.
- We also observe variations in times of the pthread implementation which may be due to memory issues/other processes already running in the cpu of the computer where the experiment was conducted.

6 Conclusion

The running times of many image processing libraries and functions such as convolution depend majorly on large scale matrix multiplication. This can be made much faster by using an optimized library such as Intel MKL [2] or openBLAS [3] can be speeded up using the multithreading concept. A combination of both can also be utilized which can speed up the running times of upcoming techniques like CNN's and applications of Computer Vision drastically to provide us with efficient and faster results.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Intelligent Signal Processing*, pp. 306–351, IEEE Press, 2001.

[2] Intel, "Intel math kernel library." Available at `https://software.intel.com/en-us/mkl`.

[3] OpenBLAS, "Openblas library." Available at `https://www.openblas.net/`.

[4] G. for geeks, "Matrix multiplication using pthreads."