COP290 Project - Imager by Prateek Garg and Rushil Gupta

1. INTRODUCTION

1.1) **What is this paper about?** Firstly, Convolutional Neural Networks is a highly growing field with lots of intricacies and potential scope of improvements. It comprises of many sub-functions like Convolution, Activation and Pooling.

In this paper, we present the modifications in pre-defined convolution execution that enhances the time complexity of the processor thereby reducing the load on the machine and generating accurate results. We incorporate use of Linear Algebra Libraries like Intel MKL and Open Blas for accelerating matrix multiplication.

1.2) **What are CNNs'?** Convolutional Neural Network (CNN) is a class of deep neural networks used for analyzing visual images and videos. CNN's usually involve little pre-processing as compared to other similar functional algorithms as they themselves learn the filters and weights without relying on the creator for the knowledge of the same. They are widely put to use in Image and Video Recognition, Recommender Systems, Image Classification, Medical Image Analysis and Natural Language Processing.
A conventional CNN design consists of an input layer, an output layer and several hidden layers. The hidden layers typically comprise of Convolution Layers, Relu Layer, pooling layers, fully connected layers and normalization layers.

1.3) Included files:
Folders : include, outputs, plots, speed_data, src, test_data

> **src** : Activation.cpp, convolution.cpp, FinalLayer.cpp, final_processor.cpp, Pool.cpp (They are the source code files of the program)
>
> **include** : Activation.h, convolution.h, FinalLayer.h, Pool.h (Include files of the program (Contains the definition of Classes))
>
> **plots** : A script for gnuplot and the plots comparing the three methods for matrix multiplication are present here
>
> **speed_data** : Data of speed observed while using pthreads, mkl or openblas
>
> **test_data** : Data used for testing is kept here, including the image and the kernels
>
> **outputs** : The compiled *.o are in this folder

Makefile : It enables quick compilation of the program via command "make". After and during compilation their object files are created in outputs folder and an executable file imager is created

readme.md : Contains the speecifications of the library and the commands used to run it

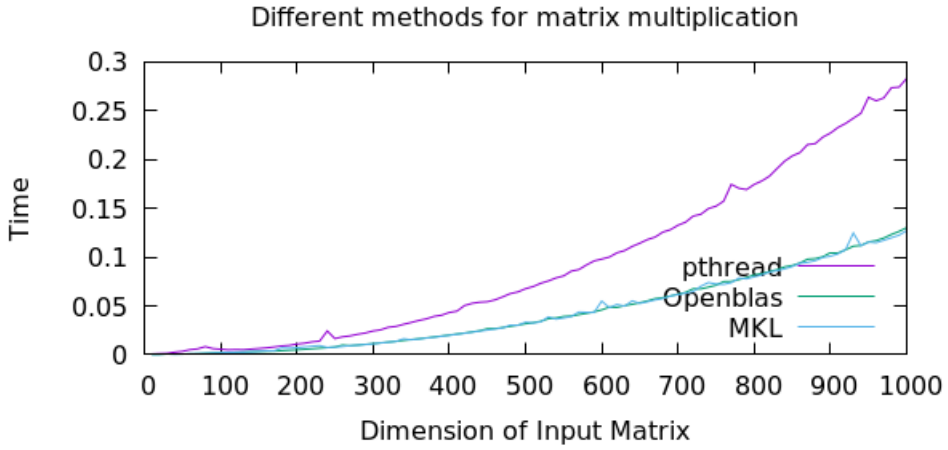
2. PROGRAM MANUAL AND DESCRIPTION

2.1) **Usage** : We haven't sent to you the executable files,so you need to run the Makefile.Just write command which is below.

2.2) **Compilation** : To use this program you must write "make" and then the name of the program ("./imager").

2.3) **Arguments** : The inputs required will be, an image and four kernels for convolution. The step by step input requirements will be printed on the terminal

3. SPEED OF DIFFERENT METHODS FOR MATRIX MULTIPLICATION



We have noticed that the speed for pthreads is the most, thus parallel computing being the slowest in this case (**num_cores = 4**). Also for different inputs generated by random number generator, we find that the speed of mkl and openblas library are the almost the same and faster than pthreads. We have noticed the time for the methods only upto 1000*1000 matrix. But we the speeds will differ for bigger matrices. Also, the speeds of these methods are dependent on the hardware on which the algorithms are running. (**OpenBlas is faster than mkl on AMD hardware**)

Point to note : mkl and openblas call the same function (have same interface) and thus they cannot be compiled in a single project as they interfere each other's working. We will have to create different projects to run them both at the same time. Though they have same interface but the inner working of both the functions are different.

The time of 3 multiplication methods is:
1) pthreads - MeanTime - 0.0939033413
Standard Deviation - 0.0846396333

2) Openblas - MeanTime - 0.0431303918
Standard Deviation - 0.0385829023

3) MKL - MeanTime - 0.0431688609
Standard Deviation - 0.038164964