Im2win: An Efficient Convolution Paradigm on GPU

Shuai Lu[1], Jun Chu[1], Luanzheng Guo[2], and Xu T. Liu[3][0000−0003−3980−9803]

[1] Nanchang Hangkong University, Nanchang, Jiangxi Province, China
`2016085400101@stu.nchu.edu.cn, chuj@nchu.edu.cn`
[2] University of California Merced, California, USA
`lguo4@ucmerced.edu`
[3] University of Washington, Seattle, Washington, USA
`x0@uw.edu`

1   ARTIFACT DESCRIPTION

Our package has 4 parts: source code (src), unit test (test), log, and gnuplot for plotting figures (plot). Please refer to the Readme file in each part in the package.

2   Software Prerequisites

CMake >= 3.10
GCC >= 7.5.0
PyTorch == 1.10.0a0
CUDA == 11.1
cuBLAS == 11.2
cuDNN == 8.0.1
gnuplot
bash
Where pytorch requires a compiled version in C++, using cublas or cuDNN to compile pytorch corresponding to different convolutional algorithm implementations.

3   Compiling and Running

The benchmark experiments used 12 different convolutions and we compared our im2win convolution algorithm with naive direct convolution, PyTorch's im2col-based algorithm using cuBLAS and cuDNN's convolution implementation on the benchmark.

3.1   How to compile.

Out of dir compilation:
$ **cd** im2win-CUDA
$ mkdir build
$ **cd** build
$ cmake ..
$ make

Or run the script:
$ bash build.sh

3.2   How to run.

The compiled test benchmark can be run using the following command:
$ **cd** im2win-CUDA
$ ./build/demo
After executing this command, different convolutional algorithms will be run on the test benchmark and performance will be recorded, with the results output to the **log** folder.

3.3   Benchmarks.

In the test experiments, we use the wall-clock time in the standard C++ library to measure the runtime of different algorithms. The batch size of each benchmark input data is 128. All tensor data is generated by functions in PyTorch.

4   Plotting

After collecting the raw data into log files, we plotted the figures as they appeared in the paper. We have provided some gnuplot plotting scripts in the **plot** folder.