Evaluating FFT-based Algorithms for Strided Convolutions on ARMv8 Architectures*

Xiandong Huang[a], Qinglin Wang[a]†, Shuyu Lu[b], Ruochen Hao[a], Songzhu Mei[a], and Jie Liu[a]

[a]National University of Defense Technology

[b]University of Pittsburgh

ABSTRACT

Convolutional Neural Networks (CNNs) have been widely adopted in all kinds of artificial intelligence applications. Most of the computational overhead of CNNs is mainly spent on convolutions. An effective approach to reducing the overhead is FFT-based fast algorithms for convolutions. However, current FFT-based fast implementations cannot be directly applied to strided convolutions with a stride size of greater than 1. In this paper, we first introduce rearrangement- and sampling-based methods for applying FFT-based fast algorithms on strided convolutions. Then, the highly optimized parallel implementations of the two methods on ARMv8-based many-core CPU are presented. Lastly, we benchmark the implementations against the two GEMM-based implementations on this ARMv8 CPU. Our experimental results with convolutions of different kernel, and feature maps, and batch sizes show that the rearrangement-based method generally exceed the sampling-based one under the same optimizations in most cases, and these two methods can get much better performance than GEMM-based ones when the kernel, feature maps and batch sizes are large. The experimental results with the convolutional layers in popular CNNs further demonstrate the conclusion above.

Keywords

CNNs, Strided Convolutions, FFT, ARMv8, Parallel Algorithm

1. INTRODUCTION

Convolutional Neural Networks (CNNs) are the most popular neural networks models in many fields. Prior works have demonstrated that the computation of the convolutional layers dominates the total overhead of CNNs. Therefore, it is essential for improving the performance of CNNs to optimize convolution operations on the specified hardware platforms.

*This research work was supported by the National Natural Science Foundation of China under grant (No. 62002365), and the National Key Research and Development Program of China (No. 2018YFB0204301).

†Corresponding author: wangqinglin.thu@gmail.com

The classic algorithms for implementing convolutions include matrix multiplication algorithm, Winograd fast algorithm and Fast Fourier Transform (FFT) fast algorithm. FFT fast algorithm is often the first choice for high-performance implementations of convolutions with large kernel, feature maps and batch sizes because of its superior characteristics. Unfortunately, current implementations of FFT-based fast algorithm only work for unit-strided convolutions with stride as 1, and cannot be directly applied to strided convolutions with a stride size of greater than 1.

While GPUs and other accelerators are the most popular platforms in deep learning fields, many factors have impelled the use of CPUs for accelerating deep learning applications [1]. For example, CPUs are still one of the most feasible platforms in leading data and super computing centers, and CPU-based deep learning tasks can further amortize their CPU-based investments. However, it's necessary for getting high performance to carefully match the architectural characteristics of CPUs and those of deep learning applications, including CNNs. In this paper, we introduce rearrangement- and sampling-based methods to utilize FFT fast algorithm for unit-strided convolutions to produce the results of strided convolutions. And we presented the details of the parallelized implementations of these two methods on ARMv8-based CPUs, which are the main computing nodes of the prototype Tianhe-3 cluster [2]. To the best of our knowledge, this is the first work which studies FFT fast algorithm for strided convolutions on ARMv8 architecture.

2. METHODOLOGY AND RESULTS

In order to get FFT-based methods for strided convolutions, we must firstly convert the strided convolutions to unit-strided convolutions. One method for converting is that strided convolutions with a stride size of s are equivalently rearranged into $s \times s$ unit-strided convolutions with smaller input feature maps and kernels. The other method is that the strided convolutions are considered as a downsampling on the output feature maps of unit-strided convolutions with the same input feature maps and kernels.

The origin implementation of the rearrangement-based method for strided convolutions is shown in Algorithm 1, and it consists of four steps. When the stride size is s, the rearrangement generates s^2 unit-strided convolutions. The specific methods of rearrangement are shown in [5].

The sampling-based method processes a common strided convolution in four steps, as shown in Algorithm 2. Compared to unit-strided convolutions, only the sampling step is

Algorithm 1: Rearrangement-based FFT algorithm for Strided Convolutions.

input : I, K
output: O^S

1 Transform I to D^r via Rearrangement and FFT
2 Transform K to G^r via Rearrangement and FFT^*
3 $Z^r = D^r \times G^r$
4 Transform Z^r to O^S via FFT^{-1}

added into the fourth step: the first step is regarding strided convolution as unit-strided convolution with the same other parameters; the second step is calculating the unit-strided convolution by FFT; the last is getting results of strided convolution from sampling the results of the unit-strided convolution.

Algorithm 2: Sampling-based FFT algorithm for Strided Convolutions.

input : I, K
output: O^S

1 Transform I to D^e via FFT
2 Transform K to G^e via FFT^*
3 $Z^e = D^e \times G^e$
4 Transform Z^e to O^S via FFT^{-1} and Sampling

In simplicity, both rearrangement- and sampling-based methods still consist of four steps: input transformation, kernel transformation, element-wise complex multiplications, and output transformation. Different from the transformations in unit-strided convolutions, these three transformations maybe not only include FFT operations, but also involve rearrangement or sampling operations. In order to achieve high performance on ARMv8 Architectures, both rearrangement- and sampling-based methods adopt the proposed optimizations in [3, 4], including the combination of scattering in transformations and packing in complex matrix multiplications, multi-level blocking, and NUMA-aware parallelization, etc.

Besides designing these two methods, we also analyze the arithmetic complexities of these two methods and compare with the direct method. The arithmetic complexities and the way to obtain them can refer to [5].

For implementing rearrangement, there are two methods: one is to rearrange the entire matrix directly, the other is to rearrange a certain part of the entire matrix according to demand. The former method requires a lot of temporary storage space and takes a lot of time. And this method not only increases a large amount of memory access, but has a negative impact on the efficiency of the operation to a certain extent as well. The latter method requires very little temporary storage space, and it takes very little time. Hence, we choose to implement the latter method. In our implementation, we rearrange a certain part of the entire matrix as needed. For example, when s=2 and the FFT tile size is 8×8, a 16×16 tile is directly obtained from the specified position of the entire matrix for rearrangement and rearranged into four tiles with size 8×8. This rearrangement method greatly reduces the amount of memory access and combines multiple matrix multiplications into one matrix multiplication, and brings certain performance improvements to the convolution operation.

Compared to FFT-based unit-strided convolutions, only the output transformations are different in the sampling-based method. We optimize the output transformations the same as the input transformation of rearrangement-based method. In other words, only the sampled results are stored back into the memory other than the whole tensor so that the memory accesses are significantly decreased.

In the experiment, we performed our implementations on Phytium ARMv8-based FT-2000plus 64-core processors, then analyzed the experimental results. The strided convolutional layers from popular networks (Overfeat, DQN, Resnet, DCGAN, All-CNN) are further used to test the relative performance of our two FFT-based strided convolution algorithms and two GEMM-based implementations. The stride size of all convolutional layers is 2.

On the layers with large input feature map and kernel sizes, our two implementations achieved better performance than MXNet-GEMM. Among them, Rea obtained the maximum speedup of 24.8 times relative to Caffe-GEMM and 17.7 relative to MXNet-GEMM. For Samp, its maximum speedup of 23.8 times was achieved relative to Caffe-GEMM and 10.9 relative to MXNet-GEMM. In summary, we got the following conclusions: our two FFT-based fast implementations for strided convolutions can get much better performance than GEMM-based implementations on the popular convolutional layers with large input feature maps, kernel and mini-batch sizes; and the rearrangement-based method can often outperform the sampling-based method in most cases.

3. REFERENCES

[1] S. Mittal, P. Rajput, and S. Subramoney, "A survey of deep learning on cpus: Opportunities and co-optimizations." *IEEE Transactions on Neural Networks*, pp. 1–21, 2021.

[2] X. You, H. Yang, Z. Luan, Y. Liu, and D. Qian, "Performance evaluation and analysis of linear algebra kernels in the prototype tianhe-3 cluster," *Asian Conference on Supercomputing Frontiers*, pp. 86–105, 2019.

[3] Q. Wang, D. Li, X. Huang, S. Shen, S. Mei, and J. Liu, "Optimizing fft-based convolution on armv8 multi-core cpus," in *European Conference on Parallel Processing*, 2020, pp. 248–262.

[4] X. Huang, Q. Wang, S. Lu, R. Hao, S. Mei, and J. Liu, "Numa-aware fft-based convolution on armv8 many-core cpus." *2021 IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2021.

[5] X. Huang, Q. Wang, S. Lu, R. Hao, S. Mei, and J. Liu, "Evaluating fft-based algorithms for strided convolutions on armv8 architectures," *Performance Evaluation*, vol. 152, p. 102248, 2021.