A method of accelerating CNN computation on DSP

Bin Li\textsuperscript{1,a}, Ying-guang Luo\textsuperscript{2,b,*} (Correspondence author), Luo Guo-bin Fu\textsuperscript{3,c}, Ye Peng\textsuperscript{4}, Jia Hu\textsuperscript{5}\textsuperscript{1,2,3,4,5}

\textsuperscript{1,2,3,4,5}Academy of Information and Communication, National University of Defense Technology, Wuhan 430010, China
\textsuperscript{a}1144029954@qq.com, \textsuperscript{b}lygsuiping@126.com, \textsuperscript{c}1339719100@163.com

Abstract—Because convolution network contains a lot of multiplication and addition operations, there is still room for optimization after it is implemented on the hardware platform. This paper mainly studies the method of accelerating the deep learning algorithm on DSP, proposes to improve the convolution layer calculation by using Winograd algorithm, and implements it on GPDSP. The final results show that it can increase the parallelism of data calculation and significantly reduce the computing time.

1. INTRODUCTION

In convolution neural network, the calculation of convolution layer takes a lot of time, so the efficiency of convolution directly affects the performance of the whole system. The convolution layer has a large number of characteristic graphs and multiplication and addition operations of convolution kernel. If we can mine the parallelism of convolution operation and reduce the times of multiplication and addition, we can reduce the calculation time.

Convolution neural network uses small convolution kernel and characteristic graph to convolute. Convolution kernel calculates with feature graph in constant moving, and there are a lot of repeated multiplication and addition operations. For example, multiplication and addition operation reaches 4G in Yolo network.

In this paper, Winograd algorithm is applied to the mapping of convolutional neural network to reduce the number of convolution multiplication and addition calculation and accelerate the implementation of deep learning algorithm[1][2].

2. INTRODUCTION OF WINOGRAD ALGORITHM

Winograd algorithm can improve the data utilization and efficiency of convolution calculation[3]. The formula of Winograd algorithm is as follows:

\[ Y = A^T \left[ (GgG^T) \odot (B^TdB) \right] A \]  \hspace{2cm} (1)

Winograd algorithm has many forms according to the convolution kernel and the output scale of one-time calculation. Taking convolution kernel as 3*3 as an example, Winograd algorithm can have different forms such as F(2*2,3*3), F(4*4,3*3), F(6*6,3*3), and the corresponding parameter matrix is also different.

F(2*2,3*3) can calculate 4 convolution results at a time, and F(4*4,3*3) can calculate 16 convolution results at a time.

F (2*2,3*3) parameter matrix is shown in Figure 1, 2 and 3. The parameters of \( f * 3 * 4 \) are shown in Figure 4, 5 and 6.
\[ A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix} \]

Figure 1 F(2 * 2, 3 * 3) a parameter matrix

\[ B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \]

Figure 2 F(2 * 2, 3 * 3) b parameter matrix

\[ G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} \]

Figure 3 F(2 * 2, 3 * 3) g parameter matrix

\[ A^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 2 & -2 & 0 \\ 0 & 1 & 4 & 4 & 0 \\ 0 & 1 & -1 & 8 & -8 & 1 \end{bmatrix} \]

Figure 4 F(4 * 4, 3 * 3) a parameter matrix

\[ B^T = \begin{bmatrix} 4 & 0 & -5 & 0 & 1 & 0 \\ 0 & -4 & -4 & 1 & 1 & 0 \\ 0 & 4 & -4 & -1 & 1 & 0 \\ 0 & -2 & -1 & 2 & 1 & 0 \\ 0 & 2 & -1 & -2 & 1 & 0 \\ 0 & 4 & 0 & -5 & 0 & 1 \end{bmatrix} \]

Figure 5 F(4 * 4, 3 * 3) b parameter matrix
For smaller convolution kernel, Winograd can have a very good effect. Winograd algorithm is used to improve the calculation method of network convolution layer. The weight and input characteristic graph are converted by Winograd algorithm and implemented on hardware platform again.

Let $r$ represent the size of convolution kernel as $r \times r$, and $M$ represent a Winograd calculation, and the output size is $m \times m$. It needs $(M+R-1)^2$ times of multiplication and addition to complete the $m \times m$ result, and the traditional algorithm needs $2MR$ multiplication and addition, which can greatly reduce the multiplication and addition operation. $F(2 \times 2, 3 \times 3)$ saves 2.25 times, $F(4 \times 4, 3 \times 3)$ saves 4 times, and $F(6 \times 6, 3 \times 3)$ saves 9 times of multiplication and addition.

3. SPECIFIC IMPLEMENTATION

Yolo is a typical convolutional neural network, which has 15 layers, including 9 convolution layers and 6 pooling layers. The convolution kernel size is $3 \times 3$ and the convolution step size is 1[4]. The Winograd algorithm can be used to improve the calculation of Yolo, and it has a significant effect.

In this paper, we use Winograd algorithm of $F(2 \times 2, 3 \times 3)$ form and $F(4 \times 4, 3 \times 3)$ form to calculate convolution and test.

By using the $F(2 \times 2, 3 \times 3)$ form algorithm, the output is $2 \times 2$ results at a time, which can reduce the calculation time to half of the original, which is consistent with the theoretical analysis.

Using $F(4 \times 4, 3 \times 3)$, when the output of the algorithm in this form is $4 \times 4$, the amount of weight data after conversion is very large, which is limited by the address space in the core. In the conversion of input characteristic graph, due to the limitation of register number, AM space is needed to store the intermediate results. The data moving outside the core consumes a lot of time, which offsets Winograd calculation, the method can reduce the times of multiplication and addition.

To sum up, we use Winograd algorithm of $F(2 \times 2, 3 \times 3)$ form to improve Yolo deep learning algorithm. It is implemented on GPDSP.

The size of Yolo convolution kernel is $3 \times 3$, and the output size is $2 \times 2$. The input matrix required for one calculation is $4 \times 4$ when calculated by Winograd algorithm.

Firstly, the weights are converted off-line, and all $3 \times 3$ convolution kernels are converted into $4 \times 4$ convolution kernels, that is, $GgG^T$ calculation is performed. The conversion of characteristic graph is carried out in the process of forward calculation in am, that is, $B^TB$. Because the parameters are relatively simple, matrix multiplication operation can be converted into addition operation. As shown in Figure 7 and 8.

The multiplication operation comes from the dot multiplication of two intermediate matrices. Finally, four convolution values are obtained by calculating the parameter matrix A.
4. PERFORMANCE ANALYSIS

After the weight and feature map are converted according to Winograd algorithm, the vector convolution calculation method is used to map on GPDSP, and the convolution calculation of each layer is carried out respectively.

Based on the original Yolo program and Winograd algorithm, the times of multiplication and addition can be reduced to 2G.

After implementation on GPDSP, the experimental results are shown in Table I. Compared with the method without Winograd algorithm, the calculation time is significantly reduced.

Finally, on GPDSP, the improved Winograd method reduces the computing time of Yolo convolution neural network to 45% of the original time.
### Table I. Calculation Schedule of Winograd Algorithm for Convolution Layer Calculation

| YOLO algorithm | Vectorization method | Computational convolution (MS) | Winograd algorithm for convolution (MS) |
|----------------|----------------------|---------------------------------|----------------------------------------|
| Conv1          | 132                  |                                 | 15                                     |
| Conv2          | 157                  |                                 | 30                                     |
| Conv3          | 81                   |                                 | 28                                     |
| Conv4          | 81                   |                                 | 29                                     |
| Conv5          | 81                   |                                 | 28                                     |
| Conv6          | 163                  |                                 | 50                                     |

### 5. Conclusion

Aiming at the typical deep learning algorithm, this method is used to improve the calculation on GPDSP, which has a good effect and can significantly improve the performance.

### References

[1] Lu L, Liang Y, Xiao Q, et al. Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs[C]// IEEE International Symposium on Field-programmable Custom Computing Machines. IEEE, 2017.

[2] Lavin A, Gray S. Fast Algorithms for Convolutional Neural Networks[J]. 2015.

[3] Clancey W J. Review of Winograd and Flores' Understanding Computers and Cognition: A Favorable Interpretation.[J]. 1986.

[4] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger[J]. 2017:6517-6525.

[5] Liu J, Chi L H, Xie L C, et al. A peak performance model for Matrix multiplication on general-purpose DSP[J]. Hunan Daxue Xuebao/Journal of Hunan University Natural Sciences, 2013, 40(11):148-152.