Contemporary Engineering Sciences, Vol. 9, 2016, no. 29, 1403 - 1411
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2016.69157

Design of a Filter Buffer Based Optimized Memory Access Method for Processing a CNN Algorithm

Kwang-Yeob Lee

Dept. of Computer Engineering, Seokeong University, 124
Seogyeong-ro, Seongbuk-gu, 02713
Seongbuk-gu, Seoul, Korea

Copyright © 2016 Kwang-Yeob Lee. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

This paper proposes an architecture that processes Convolution Neural Network (CNN) algorithm applied with Filter Buffer. The performance to be processed varies depending on the number and size of the actually used filter. Furthermore, since hardware included in the advanced driver assistance systems and smart mobile devices requires miniaturization and low power, it has a limited process element. This in turn requires an iterative external memory access. The proposed method improved the processing performance by preventing unnecessary external memory access when processing the algorithm useful to the parallelization among artificial neural network algorithms. When compared to the conventional method, the proposed method exhibited the improvement of processing performance as the memory access rate was found to be reduced by about 20% due to the application of the optimum number and size of the filter. [2]

Keywords: Convolution Neural Network, Filter Buffer, External Memory Access, Process Element, ADAS

1 Introduction

In recent years, as Machine Learning or Deep Learning has been highlighted, studies for grafting it to the computer vision technology to recognize and determine objects have actively been conducted. With regards to the image recognition technology to recognize and determine the objects through the existing images, a variety of feature point detection and recognition algorithms have been used. In particular, they have been applied in a variety of fields related

to the smart mobile devices and advanced driver assistance systems. However, due to limitations on the processing performance of simple feature point detection algorithm or recognition algorithm, Machine Learning or Deep Learning has been used. [2]

To be processed in real time, the pedestrian recognition or sign recognition included in the advanced driver assistance systems learns the applicable information in advance, and repeats the recognition and learning continuously while the system operates. The repeated recognition and learning poses limitations in performing in real time depending on the algorithm processing performance. Especially among deep learning algorithms, CNN (Convolution Neural Network) algorithm performs the convolution operation using a variety of filters, and the processing performance depending on the filter and the number of filter used influence the processing performance. [3]

In this paper, the size and total number of filters optimized when processing the CNN algorithm was presented, and the overall processing performance was improved by lowing the external memory access rate through the filter buffer.

2 Convolution Neural Network (CNN)

2.1 CNN Basic Architecture

The basic CNN algorithm is shown in Fig 1[4]. From the input image with size of n*n, the features are extracted through the repeated convolution and pooling steps. After going through the full connected step, it goes through a activation function stage, which is the activation function step, and then the classification on the input image is finally completed.

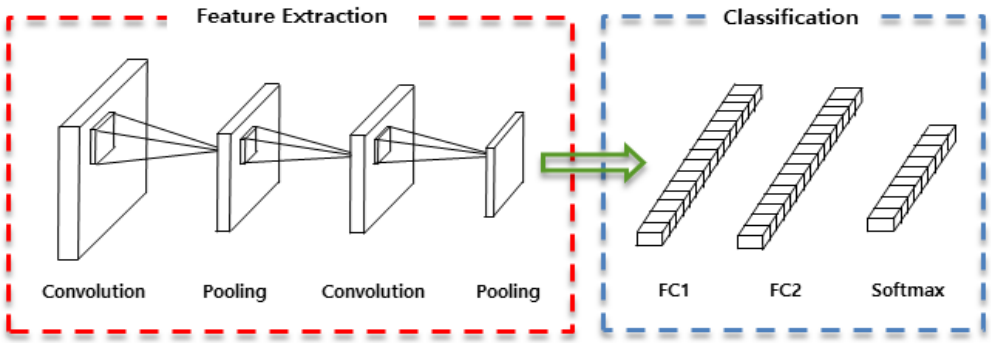

Fig 1. Interpreter execution model

One convolution step outputs m-number of feature maps using a filter with size of k*k with m-number of types (it refers to the weight in deep learning as for the filter). With the use of m-number of feature maps, more accurate recognition and classification can be made, and the parallel processing of this process can contribute to improving the processing speed.

However, in the convolution stage where the convolution operation is performed, the processing speed varies depending on the total number and size of the filter. And as the number of loading the input data and the data of the actual

filter increases due to the access to the external memory for the convolution operation, the processing speed decreases. The use of a large number of PE (Process Element) requires an increase in the amount of data loaded at a time, and therefore the number of the external memory access decreases. However, in cases where a large number of PE cannot be had due to the limited resources of hardware, as the m-number of filters with size of k*k is repeatedly loaded in the external memory to process different input data every time, the processing performance decreases. [5]

2.1.1 Convolution Layers

The basic operation of the Convolution Layer is the filtering operation of two-dimensional sliding window. The filter extracts different features, such as edges, corners and blobs. When the output values are obtained through the filter operation for each of the feature map, different filters extract different features. The values obtained through the convolution operation are normalized through a variety of activation functions. In the case of the CNN algorithm, the number and size of the filters used in the convolution operation greatly influences the performance. The following is the pseudo-code of a convolution layer and one hierarchy of the convolution layer.

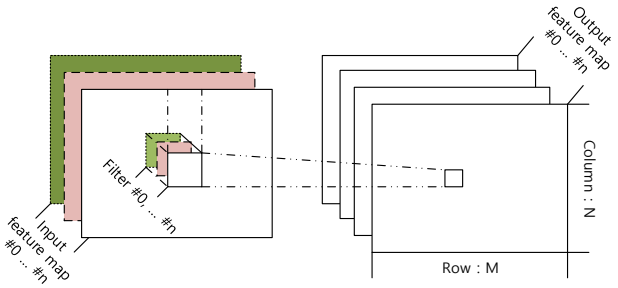

Fig 2. Convolution Layer Structure

```
PSEUDOCODE OF CONVOLUTIONAL LAYER
1 for (l = 0;l<L ;l ++){
2       for (m = 0;m<M ;m ++){
3         for (n = 0;n<N ;n ++){
4             sum = bias[l];
5             for (k = 0;k<K ;k ++){
6                   for (s1 = 0;s1 <S1;s1 ++){
7                         for (s2 = 0;s2 <S2;s2 ++){
8                sum+=weight[k][l][s1][s2] × input [k][m + s1][n + s2];
9             }}}
10        output [l][m][n] = activation_func(sum);
11 }}}
```

2.1.2 Pooling Layers

Pooling Layer reduces the number of features by summing up output values obtained from the convolution layer. This can perform two different methods like

performance and accuracy, one of which is to use the average characteristic window, which is the average pooling method, the other of which is to use the strongest characteristic window, which is the max pooling method. The number of feature maps is reduced by square times as much as the size of the window by going through the pooling layer.

2.1.3 Classification Layers

Classification layer, which is the final step of the CNN, is the class classification determination layer that generates the matter, the final input data belongs to which class, as a probability value. The output data is used to output the final result value by using the activation function as in the case of the convolution layer.

```
PSEUDOCODE OF CLASSIFICATION LAYER
1 for (i = 0;i<L;i + +){
2       sum = bias[i];
3       for (j = 0; j<K; j + +){
4               sum + = weight[i][j]  ×  input[j];
5       }
6       result[i] = activation_func (sum);
7 }
```

3 Proposed Method

3.1.1 Difference in the total number of filters used in Convolution Layer

When the first input image and data are processed in the convolution layer, the external memory access to a large number of filter operations is done. If checking the pseudo-code of the convolution operation described in Section 2.1.1, it can be confirmed that a large number of nested loops are performed. Actually, in the case of the CNN algorithm, the performance varies depending on the number of output feature maps which means the total number of filters used, and the size of filter. This is due to the time and number of loadings by accessing the filter and data used in the operation and time required for the operation itself to the external memory. Fig 3 and Fig 4 show graphs that represent the learning time and errors according to the number of filters used in the 1 layer and 2 layer of the convolution layer in accordance with the number of times for learning.

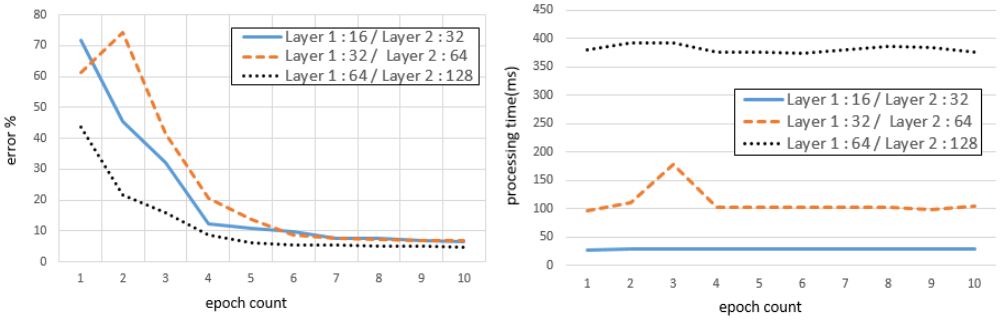

Fig 3. Error rate according to the number of Filter(Left)

Fig 4. Learning time according to the number of Filter(Right)

As the total number of filters increased, the error reduction rate was much higher as shown in Fig 3. However, it was confirmed that as the number of filters becomes larger, the operation time increases as shown in Fig 4. In fact, when 16 filters were used in 1 layer and 32 filters in 2 layers, as compared to the case where 64 filters were used in 1 layer and 128 filters in 2 layer when the epoch count was more than seven times, a narrow margin of error was shown, and the processing time decreased.

3.1.2 Difference in the size of the filter used in the Convolution Layer

As with the results according to the total number of convolution filters identified in Section 2.2.1, the size of filter also affect the processing time by the external memory access. However, it shows insufficient differences on the difference in performance compared to the change due to the total number of convolution filter. This suggests that an increase in the amount of calculation does not show a significant change on the performance compared to the external memory access according to the size of the filter in reality. Fig 5 and Fig 6 show comparisons of the operation time and error rate obtained as epoch count increases according to the size of filters used in 1 layer and 2 layer, respectively in the convolution layer.

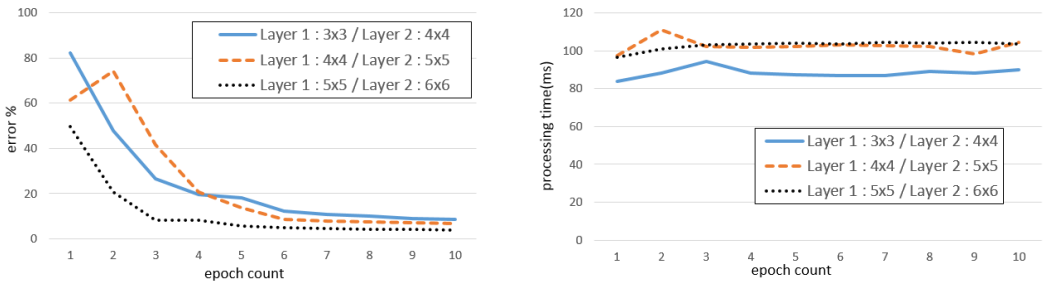

Fig 5. Error rate according to size of Filter (Left)

Fig 6. Learning time according to size of Filter (Right)

The comparison results revealed that as the total number of filters required for the convolution operation increases, the number of external memory accesses increases, which has much more effect on the processing performance than does the increase in the amount of operation due to an increase in the number of the filter.

3.2 The suggested and method of accessing memory

As identified in Sections 2.2.1 and 2.2.2, if a number filters are used, and the filter operation is performed to process the CNN algorithm, the external memory access increases. This, in effect, has a significant effect on the processing performance. The memory access method using the filter buffer architecture used in this paper helped to maintain the processing performance when a large number of filters are used in an environment with the limited PE and prevented loading the same filter again in order to obtain higher performance.

Fig 7 shows a graph that represents the number of times for the access of weight (filter) and the number of times for the access of input data according to the number of internal nodes (using filter) under the assumption that the number of PEs to be processed is four, and the size of filter is 3*3 with the input data of 640*480(VGA) resolution. It can actually be seen that as the number of internal nodes increases, the number of times for the access of weight further increases than does the number of times for the limited PE to have access to the input data.

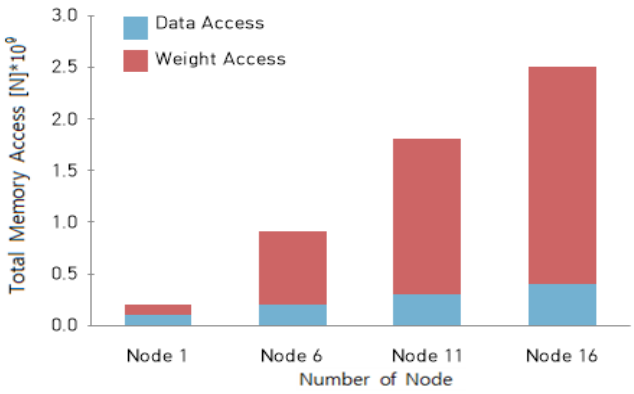

Fig 7. The number of memory accessing of data and weights according to an increase in the number of nodes

Fig 8 is a representation of the case where the filter buffer architecture is applied when the CNN algorithm presented in this paper is processed. When a large number of filters are used, repetitive memory access occurs, and if new filter is loaded in the external memory every time, it has a great influence on the processing performance. The processing performance could be improved by using the same filter after adding the buffer that can store it temporarily or preventing access to the external memory by storing the frequently used filter in the buffer in processing the CNN algorithm that has a lot of nodes with a small number of PEs.

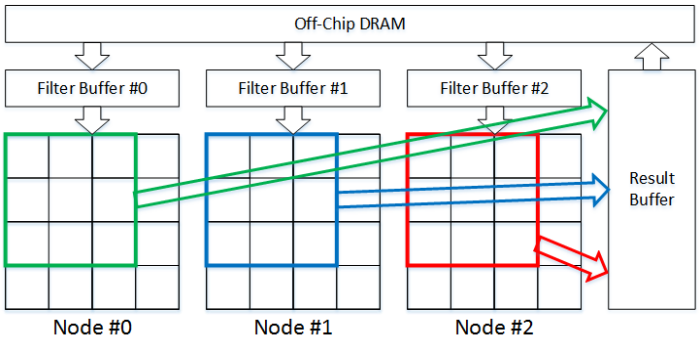

Fig 8. The access of external memory using Filter Buffers

4 Experiments and Results

Verification was conducted on the structure to process the CNN algorithm applied with the filter buffer, which is the architecture proposed in this paper. For the verification, the architecture applied with SIMD (Single Instruction Multiple Data) structure implemented in Verilog HDL was implemented, and the number of times for external memory access of the filter according to the number of nodes and the input data with size of 640*480 were simulated and compared. For the input data, the German Traffic Sign Recognition Benchmar(gtsrb) was adopted to recognize any of the signs, and the case of access to the external memory was compared in a comparison between the method of applying the filter buffer and the conventional method. The results are shown in Fig 9.

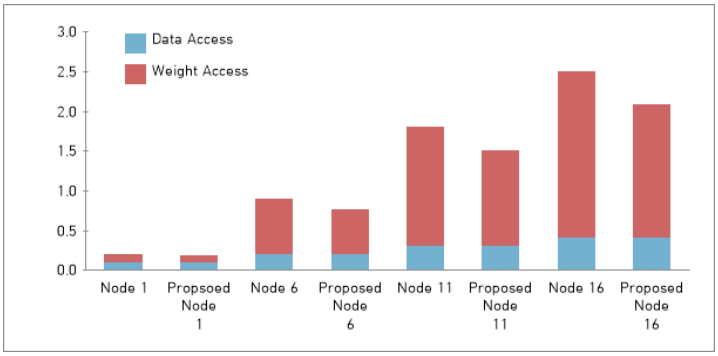

Fig 9. Comparison of the result

It can be confirmed that the number of times for loading the filter required for the filter operation further decreased compared to the number of loadings for accessing the input data to the external memory by actually applying the filter buffer. This indicates that in the case of the filter when calculated in the convolution operation or later layers, the external memory access occurs more frequently as the total number of filters increases, rather than the size of filter. In effect, it can be confirmed that the external memory access has a great influence on the processing performance in processing the CNN algorithm.

5 Conclusion

This paper proposed the architecture that processes the CNN algorithm applied with Filter Buffer. In the case of the filter used in processing the CNN algorithm, if a large number of nodes are added in order to improve the recognition rate in a situation where the total number of filters or the limited PE is had, rather than the size of the filter, a great many external memory accesses for using multiple filters are required. This significantly affects the performance on the external memory access, rather than the change in the performance due to the size of the filter. However, it can be confirmed that when the CNN algorithm is processed with the architecture applied with Filter Buffer proposed in this paper rather than the conventional structure, the actual number of times for external memory access decreases, and thus the processing performance increases by 20%. At present, since many algorithms for recognition, such as machine learning and deep learning are vulnerable to the external memory access, the possibility of degradation in processing performance may exist. However, it is expected that the improvement of performance will be achieved by solving these problems through various researches.

Acknowledgements. This research was supported by Seokyeong University in 2014.

References

[1] Kwan-Ho Lee, Jun-Mo Jeong, Jong-Joon Park, Design of a Processing Structure of CNN Algorithm using Filter Buffers, *Advanced Science and Technology Letters*, **129** (2016), 37-41. http://dx.doi.org/10.14257/astl.2016.129.08

[2] H. Deng, G. Stathopoulos, C. Y. Suen, Applying Error-Correcting Output Coding to Enhance Convolutional Neural Network for Target Detection and Pattern Recognition, *20th International Conference on Pattern Recognition (ICPR)*, (2010), 4291-4294. http://dx.doi.org/10.1109/icpr.2010.1043

[3] S. A. Dawwd, The multi 2D systolic design and implementation of Convolutional Neural Networks, *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, (2013), 221-224. http://dx.doi.org/10.1109/icecs.2013.6815394

[4] HTML Standard, Convolutional neural network. https://en.wikipedia.org/wiki/Convolutional_neural_network

[5] V. Gokhale, J. Jin, A. Dundar, B. Martini, E. Culurciello, A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks, *2014 IEEE Conference on*

Computer Vision and Pattern Recognition Workshops (CVPRW), (2014), 696-701. http://dx.doi.org/10.1109/cvprw.2014.106

[6] K. Ovtcharov, O. Ruwase, J. Y. Kim, J. Fowers, K. Strauss, E. S. Chung, Accelerating Deep Convolutional Neural Networks Using Specialized Hardware, Microsoft Research, 2015.

Received: June 15, 2016; Published: October 14, 2016