European Journal of Molecular & Clinical Medicine
ISSN 2515-8260 Volume 7, Issue 4, 2020

# Fast Convolution units for Convolutional Neural Network

[1]M.V.Nageswara Rao, [2]A.Sudhakar

*Professor, Department of ECE, GMR Institute of Technology, Rajam, India.*
nageswararao.mv@gmrit.edu.in
*Associate Professor, Department of ECE, GMR Institute of Technology, Rajam, India.*
sudhakar.a@gmrit.edu.in

***Abstract— Convolution neural network is basically an algorithm which is used to train a model to perform a specific task, which is inspired by the working of the human brain. Nowadays, its being extensively used in various applications. Its implementation over the FPGA board not only reduces the development time, but also increases the programmability. In this paper, Fast Convolution unit is described, which are based on parallel fast finite impulse response algorithm (FFA). This paper also includes the derivation of the algorithm.***

***Index Terms— Convolution neural networks, Rectified Linear Unit, Parallel fast finite impulse response algorithm, Polling Layers, Fully connected layers, DCCN, FPGA.***

## 1. Introduction

Today, Neural networks are being used for various applications like text classification, information extraction and sales forecasting. For image detection and recognition, and voice recognition, Convolution neural networks (CNN) are mostly preferred. When it comes to the mobile and real time systems, the system should be able to perform the image and voice recognition with high accuracy and low power on the embedded processors [1]. However, the CNNs which are deep enough, recognize with good accuracy. But the existing systems are not well optimized for the Deep CNN. As of now, the software CNNs are executed with respect to GPUs. But, for the mobile applications, GPU based realizations are infeasible due to tightly bounded energy consumption and limited hardware resources [2]. In the case constrained energy source, efficient architectures such as Application Specific Integrated Circuit (ASIC) and Field programmable gate array (FPGA) are used to get good performance. The threats that are to be faced in implementing high performance CNN under real time environment are: i)Requirement of large storage space-Multiple and huge in size activation maps with numerous parameters. ii)Need to implement do intricate computations -reduces the speed of training process. iii)Results in huge delay in data transfer due to less memory bandwidth. These factors also limit the performance of Deep Convolutional Neural Networks greatly, when they are implemented in real time environment with inadequate availability of Embedded systems.

This paper presents the algorithm for root of 3-parallel fast finite impulse response (FIR) algorithm and the simulation of 3-parallel fast convolution unit with 3×3 kernel. This idea can also be extended to develop convolution with larger kernels such as 5×5 and 7×7.[3] With the help of these FCUs, output can be computed in parallel and the throughput can be increased considerably.

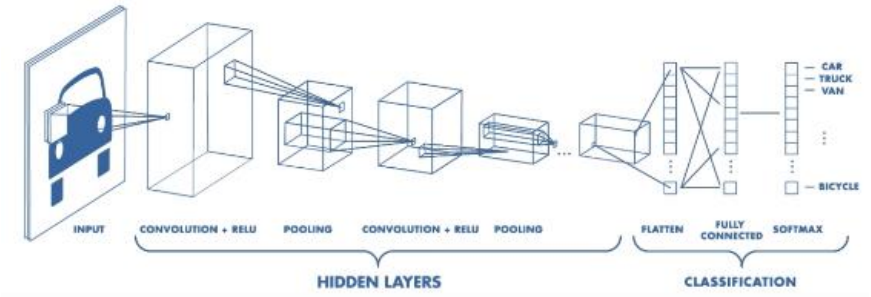


Fig. 1. A CNN Model

## 2. BACKGROUND

Neural networks are a class of algorithms, used to train a system to perform a task, without the need of being explicitly programmed. As one neuron is linked to other neuron though hidden layers, implementation of neural networks becomes very complex [4]. To decrease the complexity, convolutional neural networks were introduced. In this, the neurons are not directly connected to the hidden layers, but a convolution process takes place at the input, and only the activation map, which is generated as a result, is fed as input to the next hidden layers. The activation map generated has reduced size and parameters [5]. The convolution process is be subsequent to stimulation(activation) function and pooling layer. The activation map mostly utilizes Rectified Linear Unit (ReLU). The job of stimulation function is to decide whether the neuron in the network should get activated or not. At that point a pooling layer is utilized to reduce portrayal spatial size, number of parameters and calculation.

### 2.1 Convolutional Layers

The CNN as many convolutional layers. Each of the layer is meant to identify a given activation. If an image of a car is to be identified, it happens in many parts. Each time the convolution takes place, a kernel or filter (which is nothing but a matrix itself) traverses around the input pixels and the element wise multiplication takes place. Then All the product values are then added together to produce a single value, which is then used to produce an activation map. The convolution process is be subsequent to stimulation functions such as a sigmoid, tanh or ReLU.

### 2.2 Pooling Layers

Preserving essential features, to lessen the size of the activation map pooling layers are used. There are basically two types of pooling used.[6] One is the max pooling and other is the average pooling. Average pooling involves calculating the average value of the small local field. Whereas, the max pooling involves in picking the maximum value among all the values in a small local field. The output activation map size is made equal to the size of the input activation map is done in pooling layers.

### 2.3 Fully Connected layer

Among the last few layers of the Convolution Neural Networks one will be the fully connected layers and input to it is provided by last pooling layer[7].The input to the fully connected layer is flattened to one dimension and is put up in the form of regular neural network where each is connected to other neurons in the previous layer.

## 3. FAST CONVOLUTION ALGORITHM

### 3.1 Parallel Fast FIR Algorithm

Let p & q are the inputs & outputs then,

$$p(n) = h(n) * p(n) = \sum_{i=0}^{N-1} h(i)p(n-i) \text{ where } n = 0,1,2, \cdots, \infty \quad (1)$$

where p(n) is input sequence of infinite length and h(n) comprises the coefficients of FIR filter ofc length N. The convolution represented in Eq. (1) can be expressed in z domain as

$$Q(Z)=H(z)P(z)$$
$$=\sum_{n=0}^{N-1} h(n)Z^{-n} \sum_{n=0}^{¥} p(n)Z^{-n} \quad (2)$$

The input sequence p(n), if it is divided into odd and even samples, input can be written as $P(z)=P_0+Z^{-1}P_1$ and $H(z)=H_0+Z^{-1}H_1$.Similarly the out can be written as

$$Q(Z)=Q_1+Z^{-1}Y_1$$
$$=(P_0+Z^{-1}X_1)(H_0+Z^{-1}H_1) \quad (3)$$

Where,

$$Q_0=H_0P_0+Z^{-2}H_1P_1 \text{ and } Q_1=H_1P_0+H_0P_1 \quad (4)$$

Using FFA q0 and q1 can be written in the form of 2-parallel FFA,

$$Q_0=(H_0+H_1)(X_0+X_1)-H_0P_1 \quad (5)$$

Likewise, input sequence p(n) and tap coefficients h(n) can be divided into three parts, they are

$$Q=Q_0+Z^{-1}Y_1+Z^{-2}Y_2$$
$$Q=(P_0+Z^{-1}P_1+Z^{-2}P_2)(H_0+Z^{-1}H_1+Z^{-2}H_2)$$
$$=(P_0+Z^{-1}X)(H_0+Z^{-1}Y) \quad (6)$$

where $X=P_1+Z^{-1}P_2$ and $Y=H_1+Z^{-1}H_2$

In neq.6 the terms X and Y are in 2-parallel FIR filter form and it can be calculated by using Eq. (5) recursively. Then the computation of 3-parallel FFA is as shown below.

$$Q_0=H_0P_0-Z^{-3}H_2X_2+Z^{-3}[(H_1+H_2)(P_1+P_2)-H_1P_1]$$
$$=[(H_0P+H_1)(P_0+P_1)-H_1P_1]-[H_0P_0-Z^{-3}H_2+P_2]$$
$$Q_2=[(H_0+H_1+H_2)(P_0+P1+P_2)]-$$
$$[(H_0+H_1)(P_0+P_1)-H_1P_1]- \quad (7)$$
$$[(H_1+H_2)(P_1+P_2)-H_1P_1]$$

Parallel FFAs are basically parallel processing calculations. In this, an enormous size channel is disintegrated into a couple of little sub channels and each makes short convolutions. If a prime number T is a channel size, it can be disintegrated into T sub-channels. In the proposed parallel FFA the convolution of each sub-channel is converted into multiplication. The obtained intermediate convolutional results are added to calculate various yield in parallel.

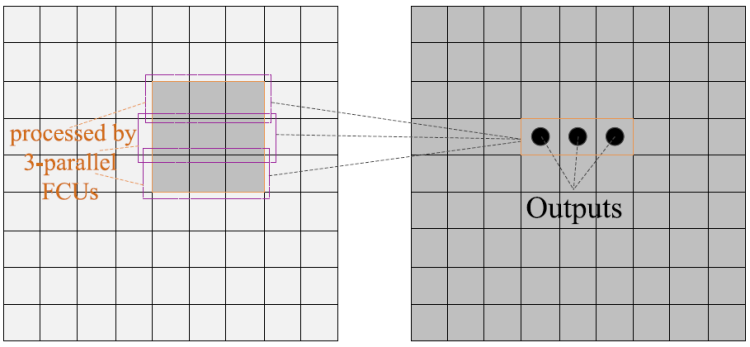


Fig. 2. Illustration of Convolution using 3×3 FCU

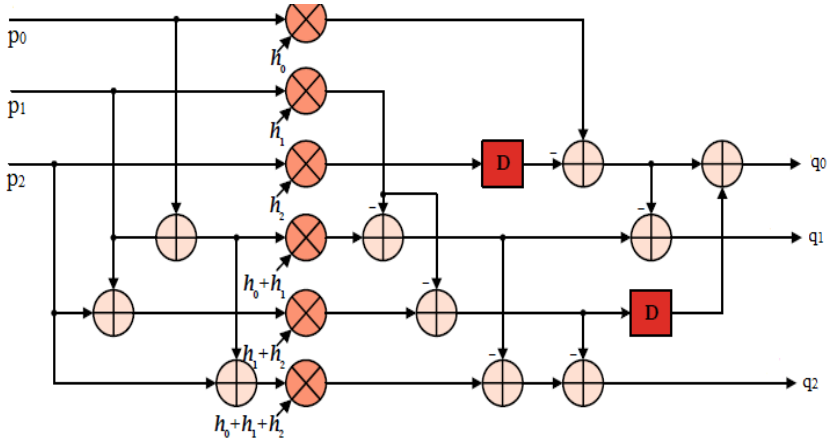


Fig. 3. Architecture of the 3-parallel FCU.

### 3.2 Fast Convolution Unit

In general, when convolution is performed using n×n kernel, then the two-dimensional convolution process is deteriorated into k one - dimensional convolutions [8]. Then, each one - dimensional convolution is performed utilizing a k tap FIR filters, where the tap coefficients are only the kernel loads.

A Fast Convolution Unit can be used to perform the convolutional process, which is based on parallel Fast FIR Algorithm (FFA). For a 3×3 kernel, a 3 parallel FCU architecture will have to be used as shown in the figure 4. Here $p_0$, $p_1$, $p_2$ are the inputs from a single row or column and $q_0$, $q_1$, $q_2$ are the outputs which are then added together to produce a single value. And the coefficients $h_0$, $h_1$, $h_2$ denote the kernel weights.
Here the kernel loads must be taken in the converse sequence because the tap coefficients made a request in each FCU is turned around and are trailed by the element-wise multiplication [9].

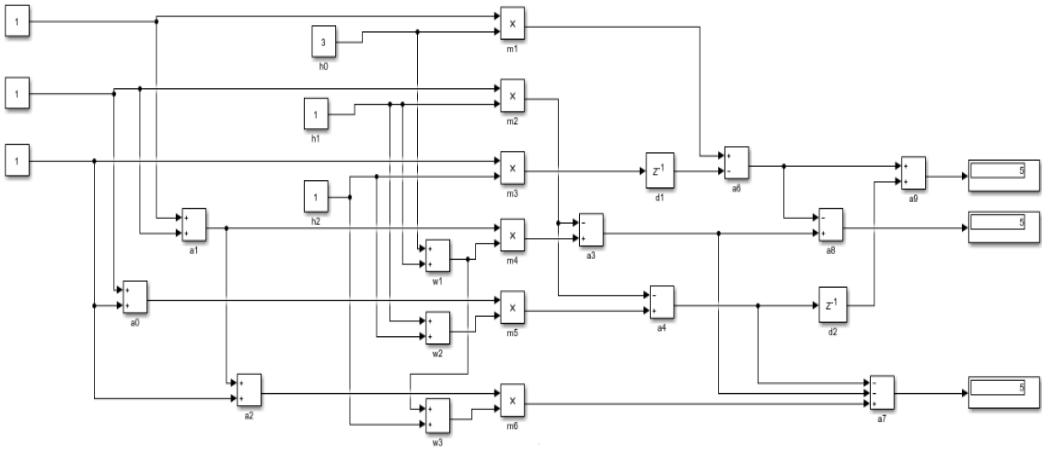

Fig. 4. Architecture of the modified 3- parallel FCU

By using this 3 parallel FCU based on 3 parallel FFA, 33% of the multiplications can be saved with some extra number of additions than the usual. If we perform 5×5 and 7×7 convolutions, using corresponding 5 and 7 parallel FCUs (which are based on 5 and 7 parallel FFA respectively), then a greater number of multiplications can be reduced [10]. The delay can be reduced further by replacing the 2 two input adders with a single three input adder. This modification increases the power by some nano watts. The modified model is shown in figure 4.

## 4. RESULTS

Implementation of Regular 3×3 with 3-parallel FCUs, Regular 5×5 with 5-parallel FCUs and Regular 7×7 with 7-parallel FCUs are carried out using Candance tool and the performance comparison for different arithmetic operations is shown in table.1.

TABLE 1: COMPARISONS OF ARITHMETIC OPERATIONS

| Convolution method | Multiplication | Addition | % of saving |
|---|---|---|---|
| Regular 3×3 | 9 | 8 | 33 |
| 3×3 With 3-parallel FCUs | 6 | 12 | |
| Regular 5×5 | 25 | 24 | 40 |
| 5×5 with 5-parallel FCUs | 15 | 35 | |
| Regular 7×7 | 49 | 48 | 43 |
| 7×7with 7-parallel FCUs | 28 | 64 | |

## 5. CONCLUSIONS

Convolution process is very essential and utilizes most of the computation power of the system. By using the FCUs mentioned above in the paper computation complexity can be reduced to large extent. This reduces the number of multiplications, hence reducing the energy requirement too.

## 6. REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun. (2015). "Deep residual learning for image recognition."
[2] A. Mollahosseini, D. Chan, and M. H. Mahoor, "Going deeper in facial expression recognition using deep neural

networks," in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Mar. 2016, pp. 1–10.
[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 580–587.
[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. (2013). "OverFeat: Integrated recognition, localization and detection using convolutional networks."
[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Int. Conf. Neural Inf. Process. Syst., 2012, pp. 1097–1105.
[7] H. Nakahara and T. Sasao, "A deep convolutional neural network based on nested residue number system," in Proc. 25th Int. Conf. Field Program. Logic Appl. (FPL), Sep. 2015, pp. 1–6.
[8] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," IEEE/ACM Trans. Audio, Speech, Language Process., vol. 22, no. 10, pp. 1533–1545, Oct. 2014.
[9] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and scalability of GPU-based convolutional neural networks," in Proc. 18th Euromicro Conf. Parallel, Distrib. Netw.-Based Process., Feb. 2010, pp. 317–324.
[10] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, "Design space explorationof FPGA-based deep convolutional neural networks," in Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC), Jan. 2016, pp. 575–580.