Efficient Implementation of CNN in Deep Learning by Using the Multirate Algorithms

Guowei Xiao
Faculty of Automation Dept, University of Technology, Guangzhou, 510006, China, 2112204008@mail2.gdut.edu.cn

Yingshuai Wang
Chendian Technology (Shanghai) Co., Ltd, tiger.wang@chendiantech.com

Ping Wang
San-xin-dui Tech, 200030, China wang@novasense-tech.com

***Abstract* — This paper proposes the multirate Convolutional Neural Networks (CNN) algorithms for an efficient implementation of the 2-Dimensional (2-D) CNN circuits implementation. During the rapid growth in computation power, Deep Learning (DL) using CNN has widened the areas of the Artificial Intelligent (AI) applications. For the layers of the convolution with pooling operation in CNN some researchers work has initially applied the multirate algorithms to the traditional (non-multirate) convolutional kernel operation of using polyphase architectures resulting in the more efficient implementation of the multirate filtering. In this work we extend it into 2-D CNN by using time-varying coefficient to achieve an efficient implementation with reduced memory(i.e. the line-buffer) size by M-fold(the pooling factor) and the MACs at 1/M of clock running rate. A design example of the first stage of CNN system will be provided. Its results are verified with the Matlab CNN-based digit recognition tool.**

Keywords—CNN; ML; DL; AI; IC; Multirate; 2-D; Signal Processing; DSP; AISC; Filter.

I. INTRODUCTION

With the surging of the computational power, Deep Learning (DL) using Convolutional Neural Networks (CNN) has become reality in more and more applications of Artificial Intelligence (AI). However, some applications have limited energy capacities. In various Internet of Things (IoT), the wearable and mobile applications of CNNs have scarce energy sources and thus require solutions to lower power consumption and smaller hardware size in order to ensure the longevity of the devices and smaller chip area [4]. As the result of the demand for lower power consumption, more research interest has been generated in exploring high-performance neural processing units or Application Specific Integrated Circuit (ASIC) accelerators with superior power efficiency and computation parallelism [5].

Fig. 1 shows a typical DL system with CNN architecture which contains convolution, pooling, and fully-connected layers. It usually includes several cascaded convolutional layers in which the a single clock frequency is employed [5].

In the applications of real-time image processing, for instance, the 2-D CNN hardware architectures that make dense, pixel-wise predictions, such as FCN [6], U-Net [7], and their variants, use very long skip lines. For example each line contains as many as 512 pixels in the U-Net image. Those skip lines are crucial for recovering of the details lost during the down-sampling. The IC hardware implementations of those networks require large memory (or line-buffer or line-delay) to store all the skip lines. The line buffers often use external memory, such as SRAM or DDR, which dramatically increases the cost in terms of silicon area footprint and consumes high power [8].

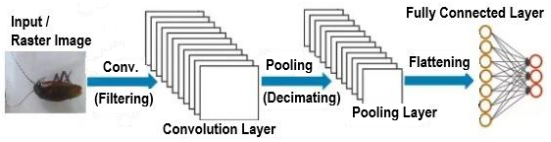


Figure 1. A typical CNN architecture with convolutional, pooling and full-connected layers

Images or 2-D signals are acquired line-by-line by the raster scan sequence. In the 2-D raster scanning-line based system, the row (vertical) delay of lines is accumulated in each convolution layer. For example, a 2x2 spatial window may cause a 1-line delay, whereas two consecutive 3x3 convolutions may result in 2-line delays and each delay contains 512 pixels to be stored in the U-Net image.

The problem with the long skip lines is that once the data on one end of the skip line is generated, it needs to be held in memory until the data in the receiving end of the skip connection is available. The more layers a connection skips over, the more line pixels need to be stored in memory. Therefore, the size of the total memory required increases with the length of the skip line. The memory requirements for the line delays can aggregate quickly and become a significant contributor to the total silicon area needed to implement the network. Moreover, the latency issues can also be problematic in latency-sensitive applications such as autonomous driving systems.

The computation of convolutional operations involves multipliers and adders, i.e., the Multiply-and-Accumulate (MAC) operation. For concurrent processing, the number of multipliers required must be the same as the filter size, which can result in large area consumption. Moreover, summing up the outputs of these multipliers involves multiple cascaded adders. Thus, digital MAC units may occupy a vast area with high power consumption [8].

The chip area and power constraint facilitate the researcher interests in the multirate filtering techniques [2] which can not only perform real-time kernel convolution but can also occupy significantly less chip area and smaller power consumption. Although the works in [2][3] are in the

analog-digital mixed-signal domain, their multirate (decimating filter) algorithms and implementation architectures can be expanded into the digital signal processing domain.

In this paper we describe the way to design decimating (multirate) filters for kernel convolution with pooling (decimating) operations, and introduce the time-varying coefficient (weight)architectures for the efficient 2-D CNN circuit implementation architectures whose memory (the line-buffer) size is to be reduced by M-fold(a pooling factor) and the MACs at 1/M of the clock rate.

The paper is organized as follows. In Section II, the multirate algorithms for 1-D decimating filter is presented in terms of time-varying coefficients. In Section III, a direct-form implementation of the 2-D CNN counterpart is derived. Finally, Section IV presents a design example of 3x3 kernel convolutional layer with pooling 2x2 (decimating filter) for demonstrating of the 2-D CNN implementation.

II. MULTIRATE ALGORITHMS FOR 1-D CONVOLUTION WITH POOLING OPERATIONS IN CNN

The multirate algorithms for efficient implementation of 1-D and 2-D filtering circuits have been previously introduced by [1][2][3] based on polyphase structures. In CNN efficient implementation, however, we modify the polyphase structures and manipulate the (decimating) filter transfer functions as filtering with the time-varying coefficients (weights) form. Thus, the resulting filter expression form is comparable to its non-multirate prototype counterpart.

The multirate algorithms in terms of time-varying coefficient expression give explicit mapping relations between non-multirate and multirate relations of z-transform functions. These can be utilized for efficient design and implementation architectures of such 1-D and 2-D decimating filters.

For the sake of easy comprehension, only the first (one) layer of CNN in Fig. 1 is discussed and illustrated. We can see that the convolutional and pooling layers architecture is the same as the 2-D decimating filtering system [2][3], as depicted in Fig. 2, with an activation operation (ReLU) which operates either after or prior to the pooling.

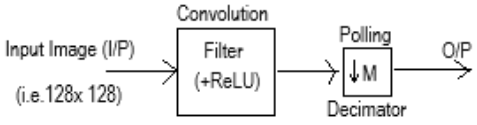


Figure 2. The convolution followed by pooling architecture can be considered as a 2-D decimating filter.

To derive the efficient implementation architecture we further consider a 1-Dimensional (1-D) linear, time-invariant (*N*-1)-th order FIR filter followed by a decimator with a factor (In neural network computation, stride for pooling layers is often used） of *M*, as illustrated in Fig. 3(a) below.

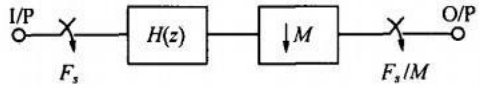


Figure 3. (a) A general filter clocking at F_s and followed by a decimation operator

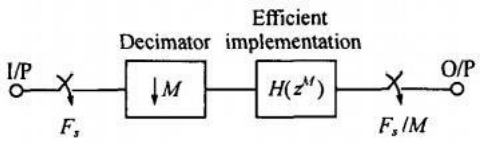


Figure 3. (b) Deriving efficient multirate implementation for an1-Dflter.

and its z-transfer function is *H(z)* as shown in Eq.1, where the unit-delay z^{-1} is related to the sampling frequency F_s . The overall system clocking at an unique frequency F_s is also called non-multirate (traditional) system,

$$H(z) = \sum_{n=0}^{N-1} h_n z^{-n} \qquad (1)$$

For an efficient implementation of Fig. 3(a) this system *H(z)* can be alternatively manipulated as Eq.2 by using multirate (decimation or interpolation) filter architecture based on the polyphase decomposition algorithms as described in [1][2].

The efficient implementation implies that the most parts in CNN operate at lower clock frequency (lower power consumption) and less memory used (smaller memory size required) especially in the 2-D and 3-D [9] CNN systems. Such a decimating filter using time-varying coefficients (weights) of convolutional layer expression can be considered as a time-variant filter with periodically varying coefficients [2][3]. It can be straightforwardly applied to CNN implementation in which the convolutional layer is followed by the pooling operation.

Considering such a decimating filtering system as in Fig. 3(a) which can be mathematically expressed as Eq.1, where the (*N*-1)-th order prototype filter with decimating factor (pooling stride) of *M*, it can be manipulated as

$$\begin{aligned} H(z) &= h_0 + h_1 z^{-1} + h_2 z^{-2} + \ldots + h_{N-1} z^{-(N-1)} \\ &= \left(h_0 + h_1 z^{-1} + h_2 z^{-2} + \ldots + h_{M-1} z^{-(M-1)}\right) \\ &+ \left(h_M + h_{M+1} z^{-1} + \ldots + h_{2M-1} z^{-(M-1)}\right) z^{-M} + \ldots \\ &+ \left(h_{N-M+1} + \ldots + h_{N-1} z^{-(M-1)}\right) z^{-(L-1)M} \end{aligned} \qquad (2)$$

where the filter order $N = ML$. Eq.2 contains L terms (and each of which contains bracketed M sequential terms that can be considered as a periodically commuted coefficient. We define such a coefficient as a time-varying one. Therefore, it has L time-varying coefficients.

Assuming $Z = (z^M)$ which is related to the reduced sampling rate F_s/M . Thus, we arrive at the transfer function with a time-varying coefficient form:

$$H(Z) = \tilde{h}_0 + \tilde{h}_1 Z^{-1} + \ldots + \tilde{h}_{L-1} Z^{-(L-1)} = \sum_{i=0}^{L-1} \tilde{h}_i Z^{-i} \quad (3)$$

where h_i represents the time-varying weights in Eq.3. It is noticed that Eq.3 has a similar math expression form to its non-multirate (prototype filter) counterpart. $H(Z)$ or $H(z^M)$ is operating at F_s/M which is a lower clock rate than the original F_s.

III. Efficient Implementation of 2-D Multirate Convolutional and Pooling Layers in CNN

Fig. 4 shows an FIR prototype 2-D filter with the transfer function $H(z_1,z)$ where z^{-1} represents the horizontal-dimensional delay unit and z_1^{-1} represents the vertical-dimensional delay unit(scan-line delay). The overall filter system is operating at the horizontal frequency F_s. This non-multirate 2-D filter can be expressed as Eq.4.

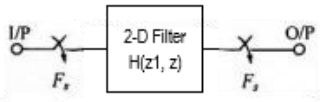


Figure 4. A non-multirate prototype 2-D filter.

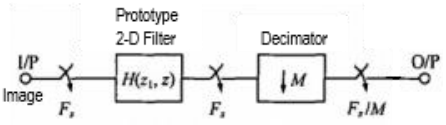
$$H(z_1, z) = \sum_{j=0}^{N-1} \sum_{i=0}^{N_1-1} a_{ij} z_1^{-i} z^{-j} \quad (4)$$

where a_{ij} are the normalized weight coefficients for both the horizontal and vertical dimensions. The index i is equal to the integer of for i = 0,1, and (N_1-1) where N_1 is defined as the filter order in the vertical dimension. Similarly, index j is for 0,1, and (N-1) where N is the horizontal dimension filtering order.

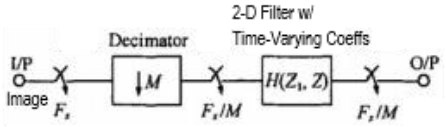
The variable separable filters (convolutions) are commonly used to design efficient neural network architectures [8]. For the demo purpose of multirate concept, assume the $H(z_1,z)$ to be variable separable. Therefore Eq.4 can be further simplified to Eq.5 [2][3],

$$H(z_1, z) = H(z_1)H(z)$$
$$H(z) = \sum_{j=0}^{N-1} a_j z^{-j} \quad (5)$$
where $$H(z_1) = \sum_{i=0}^{N_1-1} a_{1i} z_1^{-i}$$

Assuming that decimating factor M is the same in both dimensions, in Fig. 4 we apply the multirate transformation [3] to both $H(z)$ and $H(z_1)$ as similar form to Eq.3, and thus an efficient implementation can be achieved in which the scan-line memory length and computational clock speed can be reduced by a factor of M as shown in Fig. 5.



(a)A 2-D non-multirate filter $H(z_1,z)$ followed by a decimator



(b)The efficient implementation form of a 2-D decimating filter and the decimator is now in front of filter

Figure 5. Deriving the efficient multirate implementation for the 2-D filter.

Fig. 5(a) and (b) depicts the process of deriving efficient implementation of the 2-D decimating filter. It can be observed in Fig. 5(a) a typical convolutional layer followed by a pooling layer, in which the filter circuit is operating at the system maximum frequency F_s and the scan-line memory is equal to the input image pixel numbers in each line.

In Fig. 5(b), however, the decimator is placed in the front of the 2-D filter and it yields an efficient implementation when using time-varying weights. This can be described as the following Eq.6.

$$H(Z_1, Z) = H(Z_1)H(Z)$$
$$H(Z) = \sum_{j=0}^{L_2-1} \tilde{A}_j Z^{-j} \quad (6)$$
Where $$H(Z_1) = \sum_{i=0}^{L_1-1} \tilde{A}_{1i} Z_1^{-i}$$
$$Z = z^M$$
$$Z_1 = z_1^M$$

where the capital-case z_1^{-1} represents the vertical scan-line delay and the capital Z equals to (z^M), so Z^{-1}= (z^M)$^{-1}$ which implies that the computation rate (the required sampling frequency) has been lowered with the factor of M. L_1 and L_2 are the time-varying coefficient indexes, respectively. Thus, we arrive at an efficient implementation architecture of Eq.6 with the time-varying weights as shown in Fig. 6

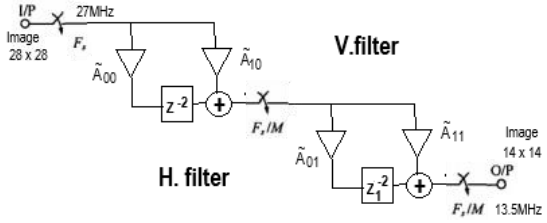


Figure 6. The proposed efficient implementation with time-varying weights

where M equals 2 in both dimensions. Thus, the time-varying coefficients can be manipulated as

$$\tilde{A}_{00} = a_0 + a_1 z^{-1}; \tilde{A}_{10} = a_2 + a_3 z^{-1}; \text{ and}$$

$$\tilde{A}_{01} = a_{10} + a_{11}z_1^{-1}; \tilde{A}_{11} = a_{12} + a_{13}z_1^{-1};$$

IV. A DESIGN EXAMPLE OF 2-D MULTIRATE CNN

Consider a 2-D FIR edge-filter example whose coefficients is listed in TABLE I below

TABLE I. THE 2-D SEPERABLE EDGE FILTER WITH BOTH DIMENSIONAL WEIGHTS

	Filter Coefficients (Weights)
Horizontal Filter $H(z)$	a_0= -3.9; a_1= 0; a_2= 4; a_3= 0;
Vertical Filter $H(z_1)$	a_{10}= -3.9; a_{11}= 0; a_{12}= 4; a_{13}= 0;

For comparison, an image as shown in Fig. 7(a) inputs to the three types of multirate (decimation) filter shown in Fig. 6.

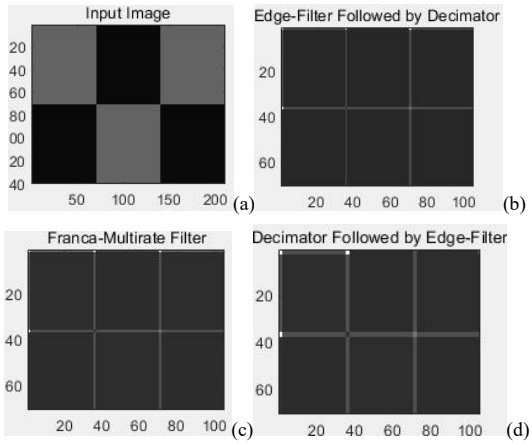


Figure 7. (a)Input image; (b)Type-I:Traditional convolution followed by pooling system's output image; (c)Type-II:Proposed multirate filter output; (d)Type-III:The pooling layer followed by the convolution layer.

The type-I is the same as convolution layer of 3x3 kernel followed by the pooling layer with stride =2 in CNN and the simulated output image is as shown in Fig. 7(b). The type-II is the proposed Franca-multirate edge filter architecture and the output image is as shown in Fig. 7(c); The type-III consists simply of placing the decimator in front of the filter and the output is shown in Fig. 7(d).

Comparing the above mentioned output images, we notice that the proposed Franca-multirate filter has the same output with the traditional convolution plus pooling's output. To further verify the multirate architecture, consider again the case of the design example for a convolution layer with pooling stride=2. The operating clock frequency is set at 27MHz. It can be seen that the entire 2-D filter now operates at a lower frequency 13.5MHz which can reduce power consumption in the circuit and the scan-line memory by half. In addition, the feature-map memory of CNN is also reduced by three quarters (image 14x14).

By using MATLAB CNN based tool at 3x3 for the digit-recognition, we compare the simulation results from the original code to modified code which models our multirate architecture in the first convolution, ReLU, and pooling layers as shown above in Fig. 5 where the bias values have been considered in the weights during the training..

The weights training has no noticeable delay or any convergence issue, and the final detecting accuracy is identical to the MATLAB original results as depicted in Fig. 8.

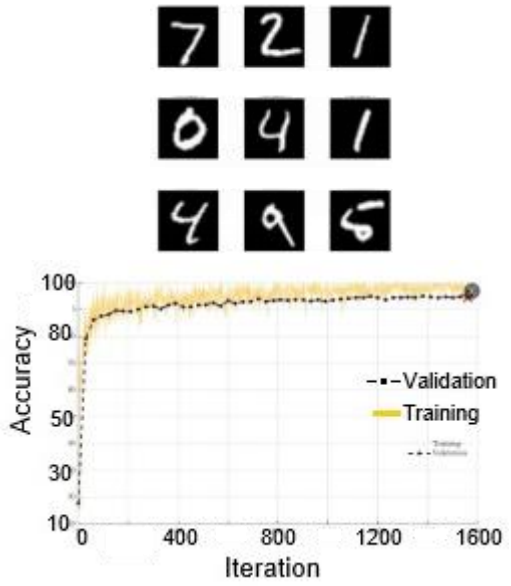


Figure 8. Digit-recognition simulation results of the multirate 2nd-order filter with pooling stride=2

V. CONCLUSIONS

We have proposed the new multirate algorithms with time-varying weight architectures for efficient CNN hardware implementation. The design example has been verified with digit-detection CNN-based MATLAB tool. It has achieved 2-fold reduction of computing clock rate and line delay memories for the CNN implementation resulting in a smaller chip size and lower power consumption.

As future work, it would be interesting to design ASIC chips to study how the efficient implementation of the while multirate CNN presented in this paper would applied into many applications in DL of AI, especially in the 2-D and 3-D CNNs. The training methodology of the multirate CNN should be further studied to achieve a similar generalized existing learning methods.

ACKNOWLEDGEMENT

The authors would like to thank previous colleagues at Instituto Superior Tecnico (IST) -Mr. Paulo Santos and

Shanghai Jiao-Tong University -Dr. Haishan Wu for their kindly reviews.

REFERENCES

[1] R. Crochiere and L. R. Rabiner, "*Multirate Digital Signal Processing",* Englewood Cliffs, NJ: Prentice-Hall, 1983.

[2] J. E. Franca,"*Non-Recursive Polyphase Switched-Capacitor Decimators and Interpolators*", IEEE Trans. Circuits Syst., vol. CAS-32, pp877-887, Sept.1985.

[3] P. Wang and J. E. Franca,"*Multirate Switched-Capacitor Circuits for 2D Signal Processing*", Kluwer Academic Publishers, ISBN 0-7923-8051-7, 1998.

[4] Y. Le Cun, Y. Bengio, and G. Hinton, "*Deep learning,*" Nature, vol. 521, pp. 436–444, May 2015.

[5] S. Gupta, "*Neuromorphic Hardware: Trying to Put Brain into Chips*.", 30 June 2019. Available online:https://towardsdatascience.com/neuromorphic-hardware-trying-to-put-brain-into-chips-222132f7e4de (accessed in 2023).

[6] Q. Chen, J. Xu,, and V. Koltun, "*Fast image processing with fully-convolutional networks*.", In Proceedings of the IEEE International Conference on Computer Vision, pp.2497–2506, 2017.

[7] O. Ronneberger, P. Fischer, and T. Brox, "*U-net: Convolutional networks for biomedical image segmentation*" , In International Conference on Medical Image Computing and Computer-assisted Intervention, pp.234–241. Springer, 2015.

[8] M. Asama, L. Isakdogan, S. Rao, B. Nayak and G. Micheal,"*A Machine Learning Imaging Core Using Separable FIR-IIR Filters*", Xiv:2001.00630v1[eess.IV] Jan 2020, Intel Co

[9] H. A. Madanayake and L. Bruton, "*A Systolic-Array Architecture for First-Order 3-D IIR Frequency-Planar Filters*", IEEE Trans. on CAS-I:, VOL.55, NO. 6, pp.1546-1559, July 2008.