Convolutional Neural Network Architecture Based on FPGA with Reduced Requirements for Parameters

P Muthu Krishnammal¹, T V Padmavathy², M Shakunthala³, and M N Vimal Kumar*³

¹Department of Electronics and Communication Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India
²Department of Electronics and Communication Engineering, R.M.K. Engineering College, Kavaraipettai, Tamil Nadu, India
³Department of Electronics and Communication Engineering, R.M.D Engineering College, Kavaraipettai, Tamil Nadu, India
Email: mnv.ece@rmd.ac.in

Abstract. The progress of deep learning has rapidly accelerated development at an exponential rate of existing technologies. Deep convolutionary algorithms have received much popularity due to exceptional success in various technology application areas. Although the quality was outstanding, their modernising has, therefore, always posed a challenge, particularly for resource-restricted hardware devices, due to their memory and computational access intensive nature of CNNs. Research article suggests a new lowered Cnn that is used for implementations for image recognition, resulting in a major decrease in the number of the communication network. Influenced by Squeezed Network, our process of reductions substitutes convolution kernel clusters with larger particles but eliminates entirely minimum duration apart from the last classification level. If implemented in software, the building model means lower computation time. By placing all qualified neural network thought-provoking Xilinx modules, we decided to utilise structure. The system design requires 2x lower complexity and a 1.2x Delay Informational processing relative to CNNs, culminating in a successful hardware design.

Keywords: Deep convolutional neural network, Current technology, FPGA.

1. Introduction

The progress of computer vision has rapidly paced the development at an exponential rate of existing technologies. Deep convolutionary algorithms, owing to their exceptional success in such a variety of software vision applications [1], have received much popularity. On the other hand, the design sophistication of the deep convolutionary human brain brings significant challenges to the seamless use of neural networks in several applications, limited by money, such as microcontrollers, and most lately, intelligent wearable tech applications and this is primarily due to the complex computing nature of CNNs and heavy storage bandwidths. For example, AlexNet architecture performs 1.46 billion processes in a single image [2-5], which requires a very large computing resource. In comparison, 60 million parameters trained by the training corpus are present in the same Alexnet[6-9]. It is impossible to store a professionally qualified network with an on storage because of the restricted storage in the latest embedded hardware platforms. However, new research developments have indicated the likelihood of reducing the number of criteria the system has mastered. Multiple experiments have been done to decrease the number of variables the system knows to mitigate both the workload...
calculation[10]. Template compressed, system coppicing and extreme encoding are all of these methods suggested in the research. Such methods of parameter reductions result in a decreased precision of the channel's identification. Investigators have made a new convolutional neural net called Squeezed Net, which has considerable criteria compared to the original Cnn model, achieves the same degree of precision, and even performs better in certain situations.

These neural Networks with little constraints mitigate the above troubles by needing less time throughout testing, less processing power and high reliability in embedded systems limited by resources like FPGAs [11-12]. We suggest a new CNN model for classification tasks in this article, using Squeeze Net, resulting in a substantial decrease in the nodes' model size. The decrease is accomplished by exchanging CNN model kernels for particles of shorter length or deleting all levels other than the last layer of fully linked classification. We achieved it using a fire node sheet, addressed in section II, to replace the hidden layer. If implemented in equipment, the proposed system resulted in less computational load. By adapting all trained neural network-on-chip using Xilinx FPGA unit, we applied the proposed design [13-15].

In contrast to LeNet, the proposed work has 11.2x fewer constraints and 2.8x improved Area-Delay material, resulting in a successful embedded system. A modern lower CNN model can be used to implement image recognition, resulting in a major reduction in the training algorithm's size. Using the quasi-simultaneous memory principle through our prior projects and integrating our architecture and design fire unit with the Xilinx FPGA system.

2. Proposed Method
CNN is a type of neural network created by a mixture of convolutionary layers, max pooling and completely linked layers that cascade. Each kernel function, defined by k, is conflated with the k X k output glass window and the next sheet. Each proposed algorithm slide by strides width and over input picture. Convolution layer conducts away back by decreasing units under one pixel with one convolutional layer. The receptive field in the convolution layers is often moved by various rows/columns, analogous to the convolutionary layer, which resulted in traits being merged by avoiding minor disturbances. In many other terms, the convolutional layers impose obstetrician by integrating characteristics into one. Deep networks even reduce the output function factor and result in fewer computing complexities for the successive layers.

![Figure 1: Convolutional Neural Network Accelerator.](image)

Driven by the Squeeze Net techniques, we used a LeNet for some of the reduction strategies. It has two convolutional, two softback layer and two completely related. The full architecture and the proportions of each sheet are seen in fig. 1. Learning these macros produces many parameters, particularly for a significant number of third-dimensional channels in a planar matrix stack.
Multiplying that set of variables in one layer by the sequence of binary filters in that providing the company all the reference value was calculated of that layer. We get a 5 X 5 kernel that results in 25 vectors of mass. The first two deep networks add 3.8% of the 430K dimensions, but the first and third entirely connected layers add, overall. However, on the other hand, one can add the amount of input x in one kernel by both the cubic of the output function dimensions of that layer to determine the number of MAC activities. The LeNet design calculates 2.1 million MAC processes, using the same estimates for the complete system. Keeping in view, as accumulating actions conduct away back, no metrics are learned, and in residual blocks, no MAC processes are needed. First, we analysed the completely related structure to decrease the number of factors in our SDN framework.

**Figure 2: Proposed method**

The first completely integrated sheet, as described previously, added 32.9% to the remainder of the parameter known. However, such a layer needs to be modified in terms of achieving substantial function decrease. We eliminated this layer in our sample provided. We covered each dot product in the design with a fire component to address the lack of precision elimination of the whole layer. We chose to substitute convolutionary layers with fire modules because convolutionary layers lead to many params. In contrast, as stated earlier, the fire module allows us by using fewer module criteria.

Both in energy usage and speed, availability of external processing is costly. As neither the sensor nodes nor the inputs data of a good system can be contained in the embedded SRAM storage, it cannot be entirely ignored. The aim is to reduce the report documenting needed and urgent public health using burst file transfers. Pre-processing is a workaround for intermittent data and disk operation linear system, but the transmission distance is large because of the restricted funds allocated in multi FPGAs.

The number of power lines is due to the average of filters, meaning that the input uses 25 times N. At least four columns should also be saved and in the input buffer and utilise all reuse possibilities. In the estimation, the outpace maps have been used by the charging process, which is a very expensive memory procedure. A tiny output buffer is used in the structure to prevent this, where each sensor channel is retained. At the same time, the values are collected from each network interface. The loading buffer, and stores all the following frame’s trained variables, is the biggest safeguard in the device. The structure here takes advantage of the fact that in each layer shown in fig. 2, the system has very few constraints, so it is not easy to recycle them all in the optimised space. The final outputs are being sent to the accumulation buffer at thread, which contains that current limit when waves are granted. Its size is greater than the inverter circuit since the present value must be stored in a column in a grid with all pooling areas.
The amount of application process is done by calculating an amount of memory in one processor by the data signal filters in that layer, equivalent to LeNet. The amount of MAC functions is determined by multiplying the number of weight vectors in a single kernel by the squares of the direction of that underlay output function. There are 4.9K variables in the last classification sheet. Our network model computes a total of 6.4M MAC functions. Therefore, contrasting the two architectures illustrates the latest design. The latest design has 6.4M operations of MAC services, while LeNet has 1.55M processes, reflecting a 4X improvement. Overall, our architect's area observation is 2.8 times greater than architecture's. It should be noted here that we will help accomplish such work to build less component architecture in building and deploying it on small hardware.

Pseudo simultaneous memory has been recommended for an acceleration hardware platforms for LeNet. The key assertion of this work is that many clock pulses can be spared through simply planning p number of control memory for each layer and spreading the input characteristics into these p memories, thereby speeding the matrix multiplication calculation. To introduce the on-chip memory location for our proposed system criteria, we have used the basic concept of the design suggested in[8]. The entire architecture adopts 8-bit encoded weight vectors or 8-bit encrypted image pixels in term of bit depth. Each layer does have a specific set of available representations, depending on the value of the core. As indicated, each layer requires k numbers of concurrent memory for just a k X processor when implementing the main memory and use a single node RAM. Growing data for each layer stores a portion of the feature vectors of such a sheet. To configure this strategy for our proposed architecture, we used a K block of memory to enforce each squee and extend layers where K will be the kernel's size. The encoding algorithms we designed in comparison is used for the sake of proper fully connected layers function.

3. Results and Discussion

The Cnn model's computation time and high resource requirements are a challenge for deployment, as described in the introduction section. In general, the implementation of CNN on different systems that are limited and hydrocarbon faces a shortage of funds to hold all the qualified variables. Therefore, as mentioned in the previous pages, if one tries to minimise CNN's qualifying criteria, it will lead to an effective upwards.

| Resources | Registers | Logic | DSP |
|-----------|-----------|-------|-----|
| Proposed  | 389       | 192618| 256 |
| Existing  | 402       | 196370| 256 |

Both architectures attain equal classification precision, as shown by the map, while our current Proposed system does this with 1.2x lower parameter. We created a single fire module of our network design from the optimum design viewpoint. Since the resources report is about the entire structure, it should be recognised that our system makes much better use of resources by concluding our resource use outcome from one fire element to the complete network design. As mentioned, the benefit of providing fewer CNN parameters is important to minimise the need for hardware resources. The trade-off of our design is the extra life imprisonment to further convolutionary operations that will occur. The average Area-Delay result for our proposed method, however, is around 2.8 times faster. This successful outcome can then be expanded and adjusted by other CNNs to achieve greater resource utilisation and hardware delivery effectiveness, as shown in Table 1, particularly for foundation embedded systems.

4. Conclusion

In the past several years, any use of CNNs in Rcmp also saw rapid growth. CNNs can provide excellent precision, but it comes at the expense of increasing magnitude in computing. Physical structures enable the rapid storage of convolutionary neural networks in tech structures to boost energy
efficiency and throughput. The paper explored different approaches to translating CNNs into Processors and showed a successful accelerator configuration for lane SoCs. It was also shown that a reasonable balance between resource needs and precision can always be made, allowing possible different equipment optimisation. There have already been studies on the resource usage of the proposed system. Our observations illustrate the regional framework performance of the proposed design. We have achieved a part of town product of 1.8 times better using our proposed methodology compared to standard LeNet architecture with the techniques outlined in this work. We intend to apply this application to the broad CNN model, mostly of the past.

References
[1]. Hadnagy, A., Fehér, B. and Kovácszázy, T., 2018, May. Efficient implementation of convolutional neural networks on FPGA. In 2018 19th International Carpathian Control Conference (ICCC) (pp. 359-364). IEEE.
[2]. Laguduva, V.R., Mahmud, S., Aakur, S.N., Karam, R. and Katkoori, S., 2020, January. Dissecting Convolutional Neural Networks for Efficient Implementation on Constrained Platforms. In 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID) (pp. 149-154). IEEE.
[3]. Hernandez, H.G.M., Mahmood, S., Brandalero, M. and Hübner, M., 2020, April. A Modular Software Library for Effective High Level Synthesis of Convolutional Neural Networks. In International Symposium on Applied Reconfigurable Computing (pp. 211-220). Springer, Cham.
[4]. Ghaffari, A. and Savaria, Y., 2020. CNN2Gate: An Implementation of Convolutional Neural Networks Inference on FPGAs with Automated Design Space Exploration. Electronics, 9(12), p.2200.
[5]. Misra, A. and Kindratenko, V., 2020, April. HLS-Based Acceleration Framework for Deep Convolutional Neural Networks. In International Symposium on Applied Reconfigurable Computing (pp. 221-231). Springer, Cham.
[6]. Yao, Y., Dong, B., Li, Y., Yang, W. and Zhu, H., 2019, July. Efficient Implementation of Convolutional Neural Networks with End to End Integer-Only Dataflow. In 2019 IEEE International Conference on Multimedia and Expo (ICME) (pp. 1780-1785). IEEE.
[7]. Brandalero, M. and Hübner, M., 2020, March. A Modular Software Library for Effective High Level Synthesis of Convolutional Neural Networks. In Applied Reconfigurable Computing, Architectures, Tools, and Applications: 16th International Symposium, ARC 2020, Toledo, Spain, April 1–3, 2020, Proceedings (Vol. 12083, p. 211). Springer Nature.
[8]. Natale, G., Bacis, M. and Santambrogio, M.D., 2017, July. On how to design dataflow FPGA-based accelerators for convolutional neural networks. In 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) (pp. 639-644). IEEE.
[9]. Xue, C., Cao, S., Jiang, R. and Yang, H., 2018, May. A Reconfigurable Pipelined Architecture for Convolutional Neural Network Acceleration. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5). IEEE.
[10]. Liu, B., Chen, S., Kang, Y. and Wu, F., 2019. An Energy-Efficient Systolic Pipeline Architecture for Binary Convolutional Neural Network. In 2019 IEEE 13th International Conference on ASIC (ASICON) (pp. 1-4). IEEE.
[11]. Thakur, C.S., Molin, J.L., Cauwenberghs, G., Indiveri, G., Kumar, K., Qiao, N., Schemmel, J., Wang, R., Chicca, E., Olson Hasler, J. and Seo, J.S., 2018. Large-scale neuromorphic spiking array processors: A quest to mimic the brain. Frontiers in neuroscience, 12, p.891.
[12]. Sharifhazileh, M., Burelo, K., Sarnthein, J. and Indiveri, G., 2020. An electronic neuromorphic system for real-time detection of High Frequency Oscillations (HFOs) in intracranial EEG. arXiv preprint arXiv:2009.11245.
[13]. Prashanth, BUV and Ahmed, M.R., 2020, January. FPGA Implementation of Bio-Inspired Computing Architecture Based on Simple Neuron Model. In 2020 International Conference on Artificial Intelligence and Signal Processing (AISP) (pp. 1-6). IEEE.

[14]. Fang, B., Zhang, Y., Yan, R. and Tang, H., 2020, August. Spike Trains Encoding Optimization for Spiking Neural Networks Implementation in FPGA. In 2020 12th International Conference on Advanced Computational Intelligence (ICACI) (pp. 412-418). IEEE.

[15]. Moreira, O., Yousefzadeh, A., Chersi, F., Cinserin, G., Zwartenkot, R.J., Kapoor, A., Qiao, P., Kievits, P., Khoei, M., Rouillard, L. and Fereouge, A., 2020, March. NeuronFlow: a neuromorphic processor architecture for Live AI applications. In 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 840-845). IEEE.