Design and Implementation of a Deep Convolutional Neural Networks Hardware Accelerator

K Sekar1*, S Gopinath2, K Sakthivel3, S Lalitha4
1Department of Electrical and Electronics Engineering, Hindustan College of Engineering and Technology, Coimbatore, Tamil Nadu, India
2Department of Computer Science and Engineering, Gnanamani College of Technology, Namakkal, Tamil Nadu, India
3Department of Computer Science and Engineering, K.S.Rangasamy College of Technology, Tiruchengode, Tamil Nadu, India
4Department of Computer Science and Engineering, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Chennai, Tamil Nadu, India
Email: *sekar.eee@hicet.ac.in

Abstract. In a wide variety of cognitive tasks, deep Convolutionary neural networks have recently demonstrated very high precision, and because of this, researchers have attracted considerable attention. Dedicated hardware semiconductors are crucial to improving their efficiency, given the enormous computation complexity of CNNs. The FPGA’s energy usage, computational strength and resilience make it a good tool for CNN additional hardware. This article suggests a coprocessor design with energy-efficient deep convolution neural networks for multi-object detection applications image processing and analysis algorithms. It can support both Convolutionary layers and fully linked layers to speed up various mobile deep learning algorithms. Also, it facilitates optimal and batches normalization activities through a separate arrangement of the pooling module. Besides, in this coprocessor, a reconfigurable activation function module serving four nonlinear functions is also realized. To draw out their connections and distinctions, we group the works into many groups. This paper is useful for artificially intelligent, hardware engineering and machine design experts, making it ideal for smart devices.

Keywords: CNNs, Deep Convolutionary network, Hardware accelerator

1. Introduction
It is a must for anyone to wear a helmet when riding a bike or two-wheeler. In general, fatalities are perhaps the most common cases in which individuals have led to imminent death due to head injury due to human brain complexity, vulnerability and different system. As we can see, in recent weeks, the policy has had to mandate everybody to wear a seatbelt. However, most rural towns still do not consider this to be a serious concern, as metropolitan cities do not have such stringent rules and procedures. This paper therefore implemented an adaptive helmet device with the introduction of the motorcycle activation circuit where even the bike starts in the absence of either the headset or a key, minimizing the probability of head trauma in the event of a collision. The existing scenario shows that with individual cars such as motorcycles and scooters, most of the people who died in crashes occurred. It should thus also provide rider with certain protection to avoid serious head injuries in unexpected cases.
2. Related Work

Field-Programmable Gate Arrays (FPGAs) have existed since the '80s. Still, they are currently becoming more and more popular due to its higher performance than software, smaller footprint and better energy-efficiency than CPUs or GPUs, and its higher-level of flexibility and time-to-market compared to custom chips (ASICs). This technology is not commonly used to implement Convolutional Neural Networks, considering all the above strengths. One of the key factors is that FPGAs are hardware programming, not software, and practitioners employed in computer vision three typically have a computer science background. They may lack the expertise necessary to programme an FPGA or explain hardware. As CNN's start to have been used to address many more difficult issues, their processing and storage specifications are growing tremendously. Traditionally, with their low bandwidth and/or efficient equipment, CNNs have been deployed on Graphics Processing Unit (GPU), but their use creates a concern. While ASIC can enable total versatility and a small form factor, it is composed of a thick design process, which is particularly important given the rapid growth of neural networks. FPGAs are attractive systems for HW 1 acceleration of CNNs due to their many appealing characteristics [1]. FPGAs typically have better energy efficiency and higher performance also than GPUs and CPUs [2]. Furthermore, the reconfigurability of FPGAs, as in the case of a framework commands processor.

Regardless of all its strengths, implementing computer vision applications on embedded systems faces a problem: Convolutional Neural Networks (CNNs) are very computationally demanding [5]. Furthermore, since they are so computationally demanding, CNNs are usually implemented on GPUs that are not energy-efficient. On the embedded systems domain, power consumption might be a constraint. Some applications may also have area limitations, requirements that GPUs or CPUs may not fulfill. Our research showed[3-4] that FPGA-based hardware offers various advantages, including mobility, region performance, reduced time-to-market, and relatively low cost, necessary for supporting compute/data-intensive applications such as CNN on embedded devices.

Even then, sparsity-induced irregularity prevents accelerators from completely exploiting the computation and reducing data. For sparse CNN models, thrilling designs on FPGAs for dense models aren't competitive because several weights are pruned such that zero operands resulting in low hardware performance are involved in most developing higher. In recent years, sparse architecture has been studied for completely layers using arithmetic operations on Field-Programmable Gate Arrays. In reality, matrix multiplication operations are the main operators of CNNs. And although spatial regularization can be translated to multipliers of the matrix-vector, this would result in a broad low memory as the input function map must be copied several times once flattened to a function. Recommends a flow of data as the basic objective exploits component multiplying. That being said, due to the extremely unbalanced load of each transport protocol, this design has a poor computational performance. Since many Look-Up-Tables (LUT) are used in this accelerator to buffer nonzero weight input activations, contributing to wasteful usage of resources can create an effective FPGA processor for sparse CNN models, the key reactions must be overcome: First of all, each output activation connects through the sliding window to multiple input installations in dense recurrent neural layers (Kernel). After pruning, the connection becomes irregular. But during compressed format, the sparse weights are encoded, resulting in additional coordinate computing to reconstruct the relation or find the performance [5-7]. However, it is difficult to design a flow of data to resolve the abnormality, thus effectively exploiting the reduction of data and computation and preserving the strong FPGA parallel processing.

While sparse CNNs have large data reductions, for complicated CNNs, it's indeed challenging to save all weights on memory. Also, there are multiple sizes of different CNN models, resulting in a high variance in the number of operations. For any CNN model, a rigid accelerated design for CNNs cannot completely exploit the limited resources of the FPGA. We rely only on organized pruning to increasing the abnormality of coarse weights to overcome all challenges. A sparse wise data flow is suggested to fix the residual abnormality of sparse weights. With the suggested data flow, we will not need additional guide computing. Also, we minimize energy by zero gratings to prevent needless calculations if the input's detections stay constant [8-10].
Figure 1: Classification of Simple CNN architecture

At the price of large amounts of computations, CNNs have experienced tremendous results in diverse applications. Standardized pruning approaches are less erratic, especially relative to unorganized pruning methods. More equipment at the cost of a significantly reduced compression rate, the weight pruning method is a successful computational and without severe precision failure. The same party will be encoded in the same compact format with shape-wise organized pruning kernels. The irregularity of sparse weights can be minimized by sharing a standard compressed format [11-15]. Also, each parallel processing engine’s load is balanced. While standardized pruning techniques minimize sparse weights’ irregularity, prior FPGA can support accelerators will still not effectively manipulate weight sparsity. We propose a sparse wise data flow and propose a selection of strategies to optimize the architecture to resolve the remaining irregularities.

3. Proposed System

The processes of manipulating MACs that have zero weights are also highly critical to miss. Recently, the research group has gained interest by modeling data flows for thin CNNs on ASIC platforms. However, due to the disparity in design between ASIC and FPGA, these dataflow would not be effective for FPGA frameworks. SCNN architecture, for example, applies input-stationary data flow where Cartesian development will be inner computing. Storage systems add a centralized indexing module to select input action potentials and then transfer selected action potentials and indexes to PEs without additional coordinate being computed. Although only performs computational tasks in measurement units by spatially swapping input activations, this data flow requires M input action potentials and N x M weights for computation, resulting in poor parallel processing on FPGAs. To speed up CNNs, we suggest a sparse, wise data flow. For a Convolutionary layer with an input function maps If amp and kernel, we pick and fetch a Toc non-zero weight vector from the Toc adjacent kernel at each access.

Similarly, we also get a vector of Tom activations in the same function map input lines shown in Figure 1. And on the PE array, we compute Toc element-vector multiplications. The coordinates of non-zero loads in various kernels are uniformed after shape-wise organized pruning. The vector hop is equivalent to the number of zeros in the same kernel row amongst neighboring nonzero weights. Convolution windows producing neighboring outputs share part of the amplification of inputs.
CNN’s are made up of several layers. Each layer is including convolution activity, batch normalization and totally connected layers. Across these levels, inputs are translated and propagated, thereby classifying or marking them. The convolution layer uses R windows to slide through the input image diagram to remove functions. The input transactions within the window are multiplied at each place by the weight matrix, and the results are summarized to determine the overall activation of the output. Note that partial sums are obtained for the measurement of output activation in various data sources.

We suggest a sparse smart data flow to bypass the purely separated computing loops that have null weights and minimize power by minimum gating to avoid wasteful computations. We suggest a Vector Converter Unit recycle or produce the required input nerve impulse for scarce CNNs. We demonstrated the acceleration architecture and loop tiling in Fig 2 to reduce off-chip memory access and improve results by chopping the input function mapping to best balance FPGA Block Interface Storage's ability.

Designers are following some good roofline model to investigate the output effects of inadequate off-chip bandwidth. The bit length of the AXI data bus that connects to the DDR is 128 bits. First of all, we do not calculate the weight and adjust the setup to find the optimum parameters for our accelerator to map the FC6 layer were shown in Figure 3. Under separate setup, Roofline model is for 8-bit into FC layer. The weight diameter is 8 bits, but until the number of PEs rises to 16, the output affects the roof. Roofline is 16-bit model fixed under various setup in theCONV layer. The quality does not change as the amount of PEs in a PU rises from 32 to 64. When the PE number reaches 8, the roof is reached by the results. Second, we're quantizing the 8-bit weights. We find that once the PE number approaches 16 and the roof becomes better the roof's output. Since the FC layer has neither mass share nor mass reuse, as the weight bit width reduces by half, each DDR access doubles the number of weights moved. So the speedup doubles as well. After that, the conve21 layer is mapped onto our accelerator to find the maximal dimensions. To obtain a high calculation efficiency, we set N as the number divider of the transmit antenna F. All production figures achieve the highest peak performance.
Restricted bandwidth is required for the low bit activations and weights, increasing performance if the acceleration is a machine design. BRAM is, however, confined to the scheduled architecture. Expanding the size of the PE array explicitly would lead to poor performance to

\[ \text{Figure 3: Block box of DDR Ram} \]

Incorporate the FPGA as the amount of BRAM needed would quickly surpass the reproductively. Thus, they can preserve the activations' data route and refine the PE layout. For example, 8-bit transfers and volumes are quantified. We dispatched two 8-bit activations picked under one PE. The two detections are repeated with much the same weight in two DSP slices, however, according to the suggested workflow. Then it is possible to concatenate and store the partial sums in the buffer. We execute output switches are connected through PUs in this work. The workload of a PU is controlled, which benefits from shape-wise pruning. The inefficiency of our architecture primarily stems from two factors when we map the network on PEs. Second, input function diagrams have zero activations, which contributes to some gated PEs. Each methodology is designed to intentionally save electricity. Second, that is not feasible to divide the feature map size uniformly by M, where N is the total number
of PEs in each PU. The DAI is dynamic and unpredictable, but dependent on the proposed transmitted data, selection of data layer, whereas the DMI relies on the suggested data flow and can be computed as the continuity equation. Sparse CNN circuits are dense CNN accelerators, and effective products are shown by the worksheet's effectiveness. We only respond to the sparsity of the weight, but we do not answer the trigger's reparability, so we test our output with DMI, which is stored in a fixed number. The efficiency is calculated for the heavy acceleration by dividing the effective bandwidth with a dynamic calculation of the network.

4. Experimental Results
The acceleration explanation is that our data flow can easily skip the sparse weight multiplications. Furthermore, this data flow flexibly maps networks to PEs, leading to high usage of on-chip capital. The output of the proposed method is shown in Figure 4. Previous research does not successfully exploit the zeros or require a poor efficiency of computation. Also, we add clock gates to unused PEs when the input activations are identical to zero. And we have greater energy efficiency relative to previous work.

![Figure 4: The simulated output of the proposed method](image)

| Table 1: Performance Comparison |
|-------------------------------|-----------------|-----------------|
| **Method**                    | **Power(mW)**   | **Performance()** |
| Proposed method               | 72.6            | 269             |
| Existing method               | 77.2            | 262             |

We align our setup with the Convolutionary FPGA accelerator in the Table1. The device architecture delivers good resource utilization performance as we exploit the sparsely and maintain maximum visualization productivity. The multiplier also optimizes durability, but the DSP performance is hindered by poor routing quality because of the unbalanced load of PE. The reduction in bit width would further increase the productivity of capital utilization. If the data width is 8-bit, the logic cell efficiency increases in the proposed system by almost 60%.

5. Conclusion
We have proposed an FPGA acceleration with such a sparse-wise data flow to circumvent zero-mass computations in this work. Also, to avoid wasteful calculations, we have used statistical data to minimize energy by zero gratings. Furthermore, for sparse CNNs, we have suggested a series of architecture optimization methods. If we achieve the sparsity of CNNs well established, the performance increase and energy consumption would be much greater we attain the sparsity of CNNs.
mentioned well. Their architecture is certainly much faster than ours, and real-time latencies are achieved. However, since our architecture's key point is its scalability and its limited usage of resources for embedded applications, it is a very unfair analogy. We want to note that they used 60 times more DSPs and more than 84 times more slices than us. In comparison, further parallelizing our architecture does not result in a relative improvement in the number of slices, only in the number of DSP units, since more of them would be concurrently computed.

References
[1] Zhang, C., Sun, G., Fang, Z., Zhou, P., Pan, P., & Cong, J. (2018). Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38(11), 2072-2085.
[2] Ma, Y., Cao, Y., Vrudhula, S., & Seo, J. S. (2017, February). Optimising loop operation and dataflow in FPGA acceleration of deep convolutional neural networks. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 45-54).
[3] Ovtcharov, K., Ruwase, O., Kim, J. Y., Fowers, J., Strauss, K., & Chung, E. S. (2015). Accelerating deep convolutional neural networks using specialised hardware. Microsoft Research Whitepaper, 2(11), 1-4.
[4] Capra, M., Bussolino, B., Marchisio, A., Shafique, M., Masera, G., & Martina, M. (2020). An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks. Future Internet, 12(7), 113.
[5] Hsiao, S. F., & Chen, Y. H. (2020, August). Flexible Multi-Precision Accelerator Design for Deep Convolutional Neural Networks Considering Both Data Computation and Communication. In 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT) (pp. 1-4). IEEE.
[6] Dinelli, G., Meoni, G., Rapuano, E., & Fanucci, L. (2020, October). Advantages and limitations of fully on-chip CNN FPGA-based hardware accelerator. In 2020 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5). IEEE.
[7] Iliev, N., & Trivedi, A. R. (2020). Low Latency CMOS Hardware Acceleration for Fully Connected Layers in Deep Neural Networks. arXiv preprint arXiv:2011.12839.
[8] Li, S., Luo, Y., Sun, K., Yadav, N., & Choi, K. K. (2020). A novel FPGA accelerator design for real-time and ultra-low power deep convolutional neural networks compared with titan X GPU. IEEE Access, 8, 105455-105471.
[9] Ye, H., Zhang, X., Huang, Z., Chen, G., & Chen, D. (2020, July). HybridDNN: A framework for high-performance hybrid DNN accelerator design and implementation. In 2020 57th ACM/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.
[10] Achararit, P., Hanif, M. A., Putra, R. V. W., Shafique, M., & Hara-Azumi, Y. (2020). APNAS: Accuracy-and-performance-aware neural architecture search for neural hardware accelerators. IEEE Access, 8, 165319-165334.
[11] Koukounis, D., Tzofis, C., Papadopoulos, A., & Theocharides, T. (2014). A high-performance hardware architecture for portable, low-power retinal vessel segmentation. Integration, 47(3), 377-386.
[12] Lim, S. J., Chae, S. H., Moon, D., Chung, Y., Lee, N., & Pan, S. B. (2009, December). VLSI architecture for the fuzzy fingerprint vault with automatic alignment module. In International Conference on Future Generation Communication and Networking (pp. 470-476). Springer, Berlin, Heidelberg.
[13] Bhairannawar, S. S., Sarkar, S., Raja, K. B., & Venugopal, K. R. (2015, February). An efficient VLSI architecture for fingerprint recognition using O2D-DWT architecture and modified CORDIC-FFT. In 2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES) (pp. 1-5). IEEE.
[14] Ren, F. (2014). A scalable VLSI architecture for real-time and energy-efficient sparse approximation in compressive sensing systems (Doctoral dissertation, UCLA).
[15] Manimegalai, C. T., Kalimuthu, K., Gauni, S., & Kumar, R. (2016). Enhanced Power Control Algorithm in Cognitive Radio for Multimedia Communication. Indian Journal of Science and Technology, 9, 25.