EAI Endorsed Transactions
on Cloud Systems
Research Article EAI.EU

Hardware Acceleration of Computer Vision and Deep Learning Algorithms on the Edge using OpenCL

B. Mishra[1]*, D. Chakraborty[1], S. Makkadayil[1], S. D. Patil[2] and B. Nallani[3]

[1]Intel Corporation, Bangalore, India
[2]Intel Corporation, Bangalore, India during the time of writing the paper
[3]Worked on the project at Intel Corporation, Bangalore, India

Abstract

Machine vision using CNN is a key application in Industrial automation environment, enabling real time as well as offline analytics. A lot of processing is required in real time, and in high speed environment variable latency of data transfer makes a cloud solution unreliable. There is a need for application specific hardware acceleration to process CNNs and traditional computer vision algorithms. Cost and time-to-market are critical factors in the fast moving Industrial automation segment which makes RTL based custom hardware accelerators infeasible. This work proposes a low-cost, scalable, compute-at-the-edge solution using FPGA and OpenCL. The paper proposes a methodology that can be used to accelerate traditional as well as machine learning based computer vision algorithms.

Keywords: CNN, OpenCL, Computer Vision, Machine Learning, Industrial Automation, FPGA, OCR, Hardware Acceleration.

Received on 08 September 2019, accepted on 02 November 2019, published on 05 November 2019

doi: 10.4108/eai.5-11-2019.162597

1. Introduction

Computer vision and machine learning enable industrial environments to become more intelligent and enable more analytics in real time. The industrial environment is very fast moving, and the large number of cameras deployed generate a huge amount of data to be processed. This data enables online as well as offline analytics. Factors such as variable latency of data transfer and data privacy make a cloud solution for such analytics unfavourable. The high speed industrial environment thus calls for application-specific compute-at-the-edge hardware accelerators to process the sensor data using, for example, computer vision algorithms.

A custom hardware accelerator has challenges of its own, including cost of the hardware, as well as time-to-market for the acceleration solution [1] [2]. Field Programmable Gate Arrays (FPGAs) have proven to be reliable accelerators for rapidly changing industries. OpenCL, which is an open source high level synthesis (HLS) framework has further helped in reducing the time-to-market of FPGA solutions for target acceleration.

This paper addresses the critical factors mentioned above for acceleration of Computer Vision based applications, especially for industrial environments. In this paper, we propose a solution methodology for hardware acceleration of Convolutional Neural Networks (CNNs) based on a combination of a Cyclone V FPGA and an Intel Atom Processor. This methodology can be also implemented to accelerate traditional Computer Vision algorithms.

Convolutional neural networks are a class of machine learning algorithms which work on multiple layers of image convolutions. This can be thought of as a cascade of feature maps from low level features, e.g. directional edges, colours, to higher level features, e.g. complex curvatures or partial regions of objects. There can various types of layers used in CNN, in this work we deal with the following:

- Convolutional layers – An NxN convolution mask that operates on the images from the input or the previous layer. Each layer has many such feature masks.

*Corresponding author. Email:bakshree.mishra@intel.com

- Pooling Layers – These are used to reduce the dimension of the images by selecting the max or average over a fixed region.
- Fully connected layers – These layers are a linear transformation of the input by a matrix multiplication.
- Activation functions – These add nonlinearity in between layers, essential for any deep neural network's operation.

CNNs are popular in deep learning based image analytics and are our target for acceleration in this paper.

Field Programmable Gate Arrays (FPGA) are re-programmable integrated circuits that can replicate hardware logic by making connections between arrays of logic gates, they also include specialized hardware that are commonly used, as well as Block RAMs (BRAM) which act like system memory. Traditionally FPGA design has been in the domain of hardware and RTL designers, but HLS frameworks such as OpenCL allow algorithms to be run on FPGAs easily. The kernel code written in OpenCL are converted to RTL by the FPGA OpenCL compilers (like Intel FPGA OpenCL compiler) and synthesizes the bit-stream. Our paper focuses on a reusable design pattern for accelerating CNNs on FPGAs which is implemented using OpenCL.

The structure of the rest of the paper is as follows. Section 2 describes related work with FPGA based CNN acceleration, Section 3 explains the problem statement. Section 4 gives the system overview for the acceleration. Section 5 and 6 describe the methodology used to accelerate traditional computer vision and deep learning algorithms using OpenCL. We present our results and performance analysis in Section 7, and summarize our work in Section 8.

2. Related Work

There are existing solutions for accelerating computer vision algorithms and deep learning networks on FPGA. Wang et al. [3] propose PipeCNN, an OpenCL based acceleration solution for CNNs that supports multiple FPGAs. However their architecture being generic, may not fit all applications in an optimal manner. We found this to be the case with our network. More about this is explained in Sections VII. Intel OpenVINO [4] is a cross platform neural network inference library that allows users to accelerate their inference on heterogeneous platforms including FPGAs. However the library currently supports larger FPGAs. Our target platform is a Cyclone V FPGA, which is low cost, low power, and is best suited for our application.

Chen et al. [5] propose the roofline method, a widely used analytical method to check the memory bandwidth and compute resources needed on a FPGA particularly for CNN architectures. Meloni et al. [6] go beyond the roofline limit to make maximal usage of the FPGA and CPU combination. Our method is reminiscent of this, as we shall see with our fully pipelined method described in Section 4. Bing et al. [7] propose an alternate method to reduce the computational load by implementing depth-wise separable convolutions.

3. Problem Statement

The target environment is an industrial setup having labels with printed text moving on a high speed conveyor belt equipped with an overhead camera. The objective is to run real time computer vision algorithms supporting high camera frame-rate. The specific use case in this paper is running Optical Character Recognition on the labels. The solution should meet the following criteria:

- Complete self-sufficiency of the solution
- Low cost solution for compute-on-edge industrial solution
- Maximal usage of CPU and FPGA at all times
- Reusable architecture for traditional Computer Vision operations as well as CNNs
- Reduced engineering efforts and faster time to market by using OpenCL
- RTL level maximal efficiency and performance extracted from OpenCL implementation

4. System Overview

The target use case is a Machine Vision application to recognize printed labels on a fast-moving conveyor belt and uses CNN to carry out Optical Character Recognition as in Figure 1.

4.1. Hardware Setup

The Apollo Island platform consists of Apollo Lake which is a Dual Core Intel Atom processor, a Cyclone V FPGA connected to the processor by a PCIe link, and a DDR3 memory. A 5 Megapixel CMOS camera is connected to the FPGA via LVDS interface. The conveyer belt is mounted with an Apollo Island based camera (consisting of Intel Atom and Cyclone V FPGA connected to a 5MP CMOS camera) to read and process printed labels.

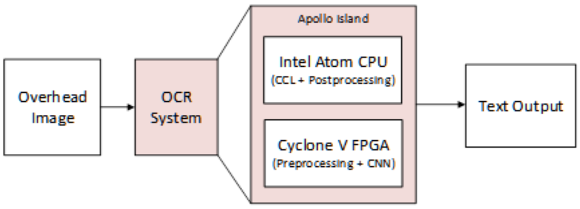

Overhead Image
OCR System
Apollo Island
Intel Atom CPU (CCL + Postprocessing)
Cyclone V FPGA (Preprocessing + CNN)
Text Output

Figure 1. Industrial Setup for fast OCR

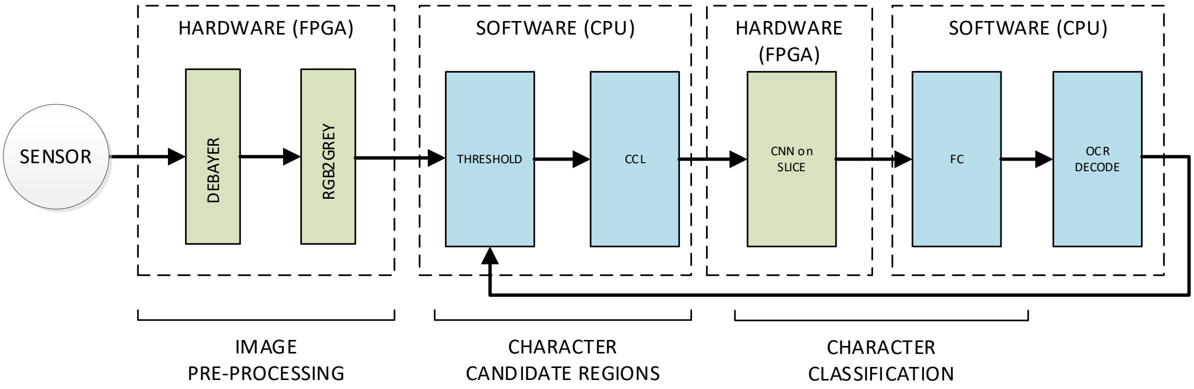


Figure 2. Industrial Setup for fast OCR

4.2. CNN based Algorithm

The stages of the algorithm are as in Figure 2. The camera on board the Apollo Island takes the overhead image of the label, the FPGA then pre-processes the image and passes it to the CPU, where connected component labelling is used to get image regions with individual characters, which are then recognized by the CNN on the FPGA, and finally the post-processing is done on the CPU to give the text output.

The FPGA pre-processes the raw image data from the sensor. The CMOS camera sensor provides raw Bayer image data. The FPGA implements de-Bayering logic to convert the raw image to RGB format. The image is then processed by RGB2Grayscale block to generate grayscale image which is passed to the CPU.

The candidate regions containing characters are generated by the Connected Component Labelling (CCL) block which is used to detect connected regions in binary images. The grayscale image is thresholded and CCL localizes and extract candidate character regions in the image. These candidate regions are then passed to the CNN sequentially.

Convolutional Neural Networks (CNN) are a class of machine learning algorithms which have recently performed very well in image classification and are very widely used for machine vision. In OCR, the input is an image and the output is a choice among a set of characters that are to be recognized. We pass the candidate character regions obtained from CCL to our CNN network instantiated on the FPGA.

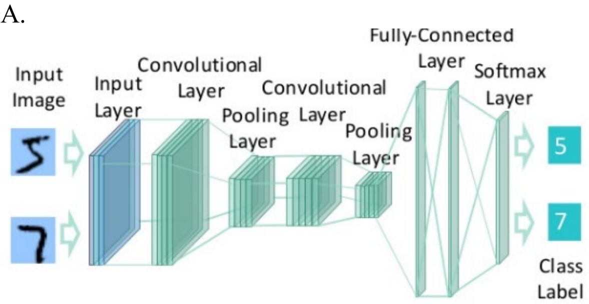


Figure 3. Industrial Setup for fast OCR

Figure 3 shows the CNN network topology trained to perform OCR in this work. The network consists of:

- Convolution layer with 16 nodes and 3x3 mask
- Pooling layer with 16 nodes and 2x2 mask
- Convolution layer with 64 nodes and 3x3 mask
- Pooling layer with 64 nodes and 2x2 mask
- Fully Connected layer with 128 nodes
- Fully Connected layer with 256 nodes

The classification in the final layer in the CNN network gives the character being recognized. The character obtained from all segmented images are then post-processed and arranged together to get the resulting text from image.

4.3. Computation Analysis

The computation calculation in Table 1 is as per the network topology described in Section 4.2. As seen in the table, the convolution operations are most compute intensive in CNN. In this paper we present an OpenCL based solution to accelerate CNN by creating custom hardware architecture to compute the convolution operations. The implementation is modular and scalable, and can be modified to suit any CNN topology.

The solution uses the dual core CPU and FPGA in a fully pipelined manner. The CPU uses two threads, one to compute the CCL, which takes the maximum amount of time, and another thread to post-process the CNN outputs and fetch new images from the FPGA. The CNN computation offload on FPGA runs parallel to these software threads. This pipelined implementation is explained in Section 5. This architecture ensures extraction of maximum resource utilization on Apollo Island platform.

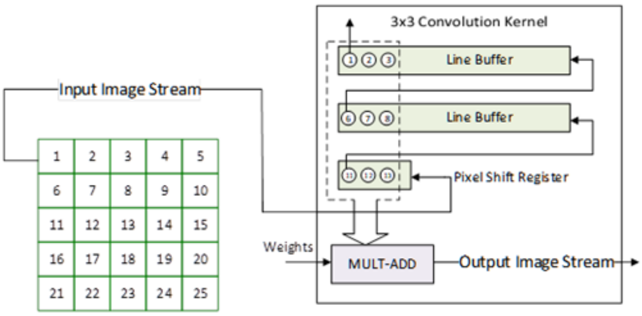
Table 1. CNN Per-Layer Compute

Layer	Nodes	Input Size	Compute
Convolution Layer 1	16	16x16	36864
Pooling Layer 1	16	16x16	4096
Convolution Layer 2	64	8x8x16	589824
Pooling Layer 2	64	8x8x16	65536
Fully Connected Layer 1	128	4x4x64	131072
Fully Connected Layer 2	256	128	32768

5. Image Convolution Kernel Model

Convolution operations, and other spatial domain filtering, require non-contiguous memory accesses, which uses high memory bandwidth. Traditional computer vision operations such as sobel, erosion, dilation share similar memory access characteristics with convolutional operation. The operations generally consist of convoluting or multiplying a mask/filter with a sub-region of an image called a sliding window. The sliding window keeps moving, allowing the operation to be replicated across the image.

The proposed hardware design pattern reads and processes an image in raster scan order. Processing image slices as a 1D data stream enables bypassing the memory fetch overhead. By using shift registers to store a maximum of N-1 rows and N pixels at a time, where N is the size of the convolution. The nodes are connected in a pipelined fashion so that each node receives an input pixel and generates an output pixel every clock cycle. This architecture is scalable to the size of the filter being utilized as well as stride, and can be utilized to accelerate both traditional as well as deep learning based computer vision.



1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure 4. Image Convolution Kernel Model

To further improve the performance, we leverage a special feature of Altera FPGAs allows the use of M10k block rams as shift registers. This dramatically enhances the resource usage in this architecture. In the Apollo Island platform, the sensor is directly connected to the FPGA, hence this architecture allows processing before the sensor even outputs the entire image, which demonstrates maximal efficiency of this architecture.

The input image size for Layer 1 in the CNN network for OCR is 16x16, and convolution kernel size is 3x3. The convolution kernel thus needs to buffer 2 rows of image data and 3 extra pixels and start processing the convolution. The raster scan architecture ensures that one pixel is processed every clock cycle.

6. CNN Accelerator Hardware Architecture

Figure 5 shows the hardware architecture for accelerating the convolution layers of the CNN network. The input image, as well as the intermediate outputs, are accessed in a raster scan order, as described in Section 5. The convolution nodes of the CNN topology in depicted in Figure 4 are implemented on the FPGA. They receive image slices with characters and process them across multiple layers and send the result to the CPU to compute the fully connected layers and perform the post processing for OCR.

- The convolution operation is a dot product of two vectors. OpenCL naively uses more DSPs than required for the multiplication operation, hence a small custom RTL block is instantiated to optimize the DSP usage. The pooling operation is an averaging of 2x2 image slices.
- Owing to limited resources available on the FPGA, the physical nodes are distributed among Layer 1 and Layer 2 to balance the computational load between the layers. A weight buffer, for storing all the network's weights, is used to reduce the CPU DDR overhead of loading many weights every cycle. A unique methodology is introduced to compute partial results of the Layer 2 convolutions, as all outputs from Layer 1 are not available to Layer 2 at the same time. The non-linear activation function is the ReLU operation.

6.1. Compute Balancing and Partials Computation

The CNN accelerator leverages a key compute balancing strategy to maximize the active usage of hardware resources. The number of nodes for different layers of CNN that are physically instantiated in hardware is determined by the number of computes as in Table 1 as well as available resources on FPGA. The nodes of a layer get processed iteratively by the instantiated physical nodes, or kernels. Two layers are connected together by a FIFO which stores the data generated by the previous layer and is accessed iteratively by the nodes of the subsequent layer. The raster scan order is maintained in the FIFO across the different outputs, or feature maps, from the nodes of the previous layer.

A compute balanced hardware consuming maximum DSPs and that is active without being idle in any clock cycle is achieved by a unique Partials Computation methodology. All nodes in one layer need to generate output feature maps for the next layer to start processing.

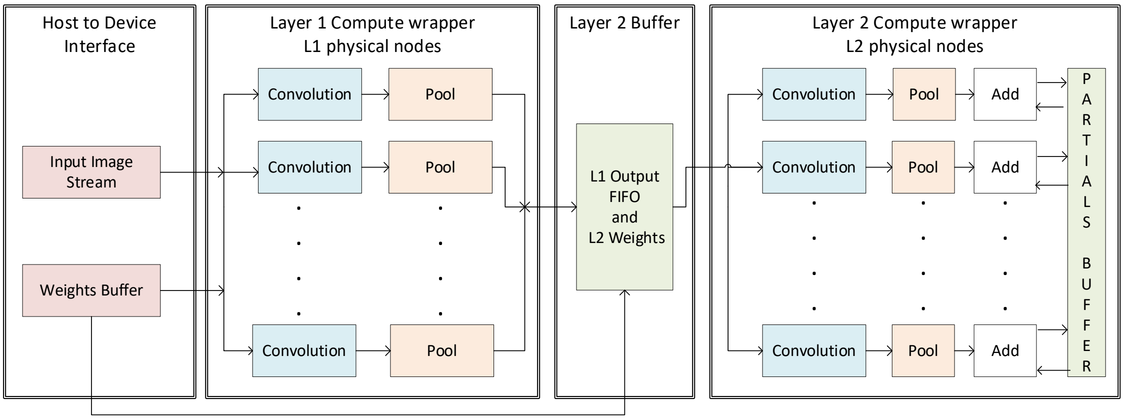


Figure 5. High Level Hardware Architecture for CNN Acceleration

This creates stalls and affects the performance of the FPGA. To address this issue, an architectural scheme is presented that enables a layer to start processing with minimal data from the previous layer by computing partials. This modification has been critical in maximizing the resource activity on the FPGA.

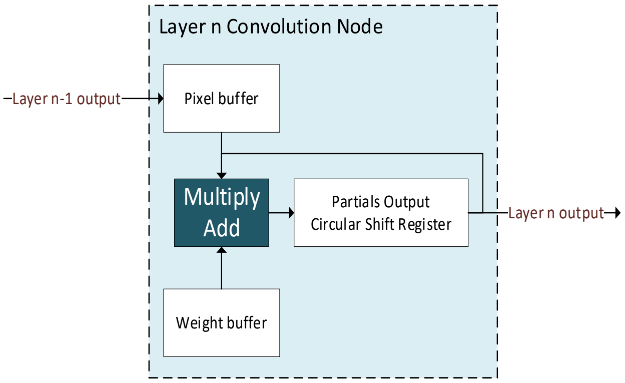


Figure 6. High Level Partial Compute Block

The computations of Layer 2 require outputs from all nodes of Layer 1. We alleviate this problem by using the fact that the output of Layer 2 can be computed as the sum of convolutions over each individual node of Layer 1. We call the results of these individual convolutions as the partials of Layer 2, these are stored in the Partials Buffer, which is a circular shift register. As the outputs from Layer 1 are computed, the partials are updated as shown in Figure 6, until all the nodes are completed. This unique architecture allows continuous convolutions without any stalls and allows the hardware to operate with maximum performance.

6.2. Weights Buffering on FPGA

CNN is a high bandwidth application which operates on huge amount of data as weights, inputs and intermediate as well as final outputs. Weights of the neural network, especially for Layer 2, need to be continuously updated as the convolutional kernels iterate many times over all the nodes. This creates a bottleneck in the memory bandwidth and slows down the input image streaming pipeline from the CPU. Hence, all the weights of the network are stored on board the FPGA in M10k blocks. This frees up the input and output streams and also saves CPU overhead of indexing different weight fetch requests.

7. Results and Performance analysis

The CNN accelerator presented in this work has been developed using OpenCL and has been optimized to meet RTL level performance. The resource area usage is as in Table 2. The different blocks present on the FPGA that we report the numbers are based on ALM (Adaptive Logic Modules), DSP (Digital Signal Processors) and M10ks, which are the atomic unit of system memory on the FPGA, equal to 10 kB of memory.

Table 2. Resource Utilization on Cyclone V FPGA

Resource	Percentage Used
ALM	88
DSP	76
M10K	44

Table 3. Resource Utilization on Cyclone V FPGA

Threaded Operations	CPU (ms)	FPGA (ms)
IO Channel (FPGA)	-	8.3
CCL+ Threshold	25	-
CNN-Conv (FPGA)	200	8
CNN - FC	15	-

We had tested the resource usage of PipeCNN on our optimized architecture, however it was unable to fit in the FPGA and ALM usage reported by the fitter was 116%. It also used 5 DSP cores per 3x3 convolutional kernel whereas our implementation uses only 4 without any loss of accuracy.

Table 3 provides profiling data for the software as well as for the hardware accelerated flow for OCR. The Cyclone V hardware operates on a frequency of 132MHz, and the end to end application processes 220 characters in 33ms.

The hardware achieves 25x performance over convolution layers. The software flow could originally compute OCR at 4 FPS and the CNN accelerator boosts the end-to-end performance by 7.5X by running at 30FPS.

The state of the art OCR implementations recognize 20 words in 350ms and 100 words in 500ms. Taking that average number of characters in a word is 4.84, the time to recognize a single character takes at least 1.033ms. The implemented architecture on the other hand takes 0.15ms to recognize a character and demonstrates 6.8x better performance.

The impact of automation is immense and deeply affects all kind of industries. Modern industry is extremely cost sensitive and looking for low cost solutions without compromising on speed of processing. Fast TTM and flexibility to change or fine tune requirements is very critical.

8. Summary

This work presents a unique architecture to accelerate Convolutional Neural Networks and spatial domain computer vision operations in general. This was implemented using OpenCL on Intel Apollo Island Platform, which is a low cost FPGA solution. OCR is an effective way decode a specific part number, date of manufacturing, date of expiry etc. in a fast moving conveyor belt and is a key machine vision application in industrial environment.

Acknowledgements.

We would like to thank our colleagues at Intel Bangalore and Intel Penang who supported this activity

References

[1] Abdelouahab, Kamel, et al. "Accelerating CNN inference on FPGAs: A Survey." arXiv preprint arXiv:1806.01683 (2018).

[2] Zhao, Wenlai, et al. "F-CNN: An FPGA-based framework for training convolutional neural networks." 2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE, 2016.

[3] D. Wang, K. Xu and D. Jiang, "PipeCNN: An OpenCL-based open-source FPGA accelerator for convolution neural networks," 2017 International Conference on Field Programmable Technology (ICFPT), Melbourne, VIC, 2017, pp. 279-282.

[4] OpenVINO - Open Visual Inference and Neural Network Optimization Toolkit, Intel Corporation, https://software.intel.com/enus/openvino-toolkit.

[5] Zhang, Chen, et al. "Optimizing fpga-based accelerator design for deep convolutional neural networks." Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2015.

[6] Meloni, Paolo, et al. "Curbing the roofline: a scalable and flexible architecture for CNNs on FPGA." Proceedings of the ACM International Conference on Computing Frontiers. ACM, 2016.

[7] Liu, B.; Zou, D.; Feng, L.; Feng, S.; Fu, P.; Li, J. An FPGA-Based CNN Accelerator Integrating Depthwise Separable Convolution. Electronics 2019, 8, 281.