Low-Power-Consumption Deep Convolutional Neural Network Accelerator for ADAS

Attention has been increasingly focused on the introduction of artificial intelligence (AI) technologies, particularly recognition and classification technologies applying deep convolutional neural networks (DCNNs), that can play a key role in realizing the safety of automobile driving operations using advanced driver assistance systems (ADAS). However, as DCNN processing involves a huge number of calculations and massive volumes of data, it is difficult to apply it to automotive systems with limited power consumption.
To overcome this problem, Toshiba Electronic Devices & Storage Corporation has developed a real-time hardware accelerator (HWA) with low power consumption for automotive applications. This HWA makes it possible to efficiently implement DCNN processing by achieving reductions in the number of memory accesses and the volume of data.

1.Introduction

The number of automobiles equipped with collision mitigation braking and other ADAS is increasing to realize safe driving operations. Just as human drivers mainly rely on visual perception to drive an automobile, camera-based ADAS use images to recognize objects around an automobile. Therefore, the use of image recognition is essential to enhance the functionality of ADAS.

The capability of image recognition is dramatically increasing because of the recent progress of artificial intelligence (AI) technologies incorporating deep learning, particularly the application of deep convolutional neural networks (DCNNs) specifically designed for image recognition. DCNN processing is now attracting a lot of attention as a means of realizing high-accuracy and sophisticated ADAS. However, since DCNN processing involves a huge number of calculations and massive volumes of data, it incurs an increase in power consumption. For example, a graphics processing unit (GPU), which is widely used for the research of DCNN processing, consumes more than 100 W. On the other hand, to achieve high reliability, automotive systems require semiconductor devices capable of operating under harsh temperature conditions without cooling fans, which are susceptible to high failure rates. It is therefore necessary to realize DCNN processing with low power consumption in order to reduce the amount of heat generation.

To meet this requirement, Toshiba Electronic Devices & Storage Corporation has developed a hardware accelerator (HWA) and a dedicated tool. This HWA (hereinafter referred to as “the DCNN HWA”) efficiently performs the DCNN inference processing and achieves a reduction in power consumption by reducing the number of memory accesses and the amount of memory read and write data. The DCNN HWA realizes real-time DCNN processing with low power consumption, eliminating the need for a cooling fan. This report describes an overview of the newly developed DCNN HWA and the dedicated tool.

2. Application of DCNN processing to ADAS

Figure 1 shows an example of application of DCNN processing to an ADAS that provides semantic segmentation, i.e., image classification at the pixel level. In this example, the DCNN HWA recognizes, at the pixel level, what is in an input image from a camera, including road lanes, pedestrians, buildings, and other objects, and labels them in different colors. While conventional image recognition techniques detect road lanes by identifying white lines that indicate sidewalk-roadway boundaries, DCNN processing is capable of detecting road lanes without these white lines. Since DCNN processing provides more advanced and sophisticated image recognition with higher accuracy, there is strong demand for its application to ADAS to enhance their functionality.

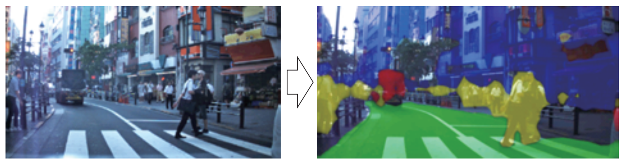

Figure 1. Example of semantic segmentation using DCNN
High-accuracy, sophisticated image recognition using DCNN processing has a capability to realize detection of road lanes even when a road has no white lines that indicate sidewalk-roadway boundaries.

3. DCNN HWA

DCNN processing consists of many network layers as shown in **Figure 2**. In particular, convolution and fully connected layers require massive multiply-accumulate (MAC) operations. In addition, MAC calculations process an enormous amount of data (including input image data, intermediate data between network layers, and weight data), requiring a huge memory bandwidth. For example, the inference processing of VGG-16(1), an image recognition network, needs roughly 15.5×10^9 MAC calculations and 138×10^6 weight data for each image area to be recognized.

Our DCNN HWA and the DCNN HWA configuration generator tool are designed to achieve real-time DCNN processing with low power consumption.

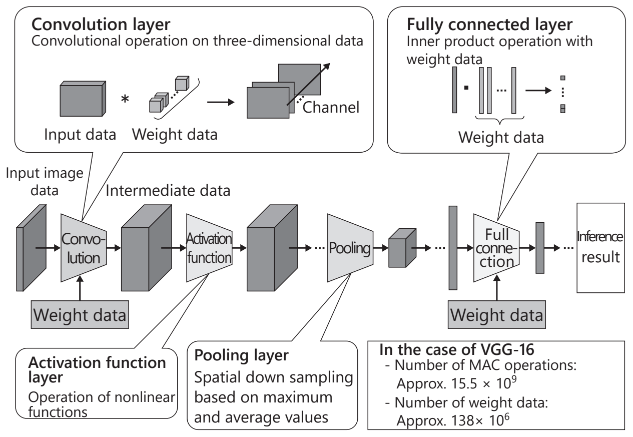
Convolution layer
Convolutional operation on three-dimensional data
Input data
Weight data
Channel
Fully connected layer
Inner product operation with weight data
Weight data
Input image data
Intermediate data
Convo-lution
Activation function
Pooling
Full connection
Inference result
Weight data
Weight data
Activation function layer
Operation of nonlinear functions
Pooling layer
Spatial down sampling based on maximum and average values
In the case of VGG-16
- Number of MAC operations: Approx. 15.5 × 10^9
- Number of weight data: Approx. 138× 10^6

Figure 2. Typical architecture of DCNN
DCNN processing consists of many network layers and requires huge amounts of data and MAC operations.

3.1 Architecture of the DCNN HWA

Figure 3 shows the architecture of the DCNN HWA. The DCNN HWA is composed of: (1) a direct memory access (DMA) unit that writes output data to and read input data from an external dynamic random access memory (DRAM); (2) an internal 1 Mibyte (Mibyte: mebi (2^{20}) bytes) memory that holds input, output, and intermediate data; (3) an execution unit that performs major computing operations of DCNN processing; (4) a control unit that provides overall

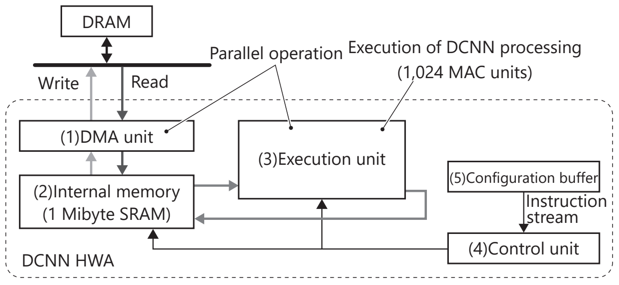

Figure 3. Architecture of HWA for DCNN processing
Parallel operations of the execution and DMA units improve the efficiency of DCNN processing.

control; and (5) a configuration buffer that holds control sequences.

The DMA unit transfers data between the internal memory and an external DRAM while the execution unit is performing computing operations. Such parallel operations help reduce data wait states and increase the utilization of the execution unit, improving the efficiency of DCNN processing.

The DCNN HWA performs computing operations while accessing the internal memory, which is composed of static RAMs (SRAMs) that consume less power than DRAMs. However, it is still important to improve the efficiency of the internal memory because a huge number of memory accesses cause large power consumption.

To overcome this problem, the execution unit is designed to reduce the number of accesses to the internal memory. The execution unit processes data from four channels in parallel using four processing elements (PEs) as shown in **Figure 4**. Since the input data to the convolution layer from all the four channels are identical, the execution unit reduces the total number of internal memory accesses by feeding the same input data from the internal memory to four PEs. In each PE, the computing circuits that perform frequently used operations in each layer are connected in cascade in order to eliminate the need to write and read intermediate data between layers into and out of the internal memory, thereby reducing the number of internal memory accesses. Furthermore, the convolution and fully connected layer units in the PEs incorporate a total of 1,024 16-bit floating-point MAC units (256 MAC units per PE) to perform a huge amount of MAC operations in real time.

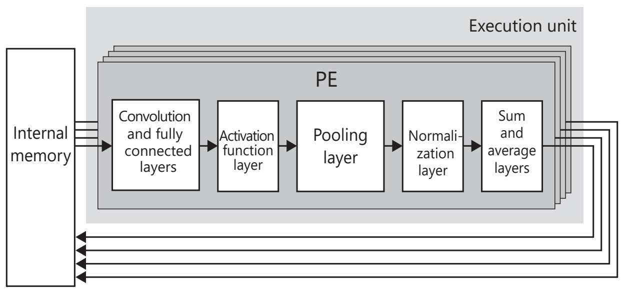

Figure 4. Architecture of execution unit

Major operation units for DCNN processing are connected in cascade to reduce the number of accesses to the internal memory.

3.2 DCNN HWA configuration generator tool

We have developed the DCNN HWA configuration generator tool to convert the models pre-trained on a PC so as to facilitate porting of the models to the DCNN HWA. **Figure 5** shows the DCNN processing flow using the DCNN HWA configuration generator tool. This tool takes, as input, the network definition of Caffe[(2)], an open-source deep-learning framework, and weight data and generates a binary executable for the DCNN HWA (DCNN HWA configuration binary data). The binary DCNN HWA configuration data consist of the settings to control the DCNN HWA and weight data converted into the DCNN HWA format. The DCNN HWA receives the DCNN HWA configuration binary data and input image as input, performs DCNN inference process and generates inference results.

The DCNN HWA configuration generator tool optimizes the amount of data read and written from/to external DRAM by: (1) partitioning a network into smaller sub-networks so that all the data necessary for an operation fit in the 1 Mibyte internal memory of the DCNN HWA; and (2) performing operations for multiple layers in an integrated manner (sub-network integration). These processes are explained below:

(1) Network partitioning

The huge amount of data required for DCNN processing does not fit in the 1 Mibyte internal memory of the DCNN HWA. To solve this problem, the DCNN HWA configuration

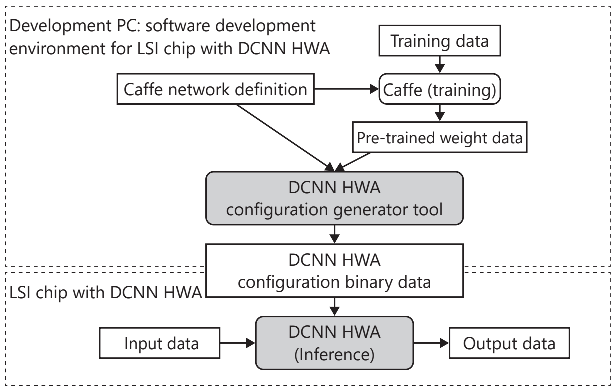
Development PC: software development environment for LSI chip with DCNN HWA
Training data
Caffe network definition
Caffe (training)
Pre-trained weight data
DCNN HWA configuration generator tool
DCNN HWA configuration binary data
LSI chip with DCNN HWA
Input data
DCNN HWA (Inference)
Output data

Figure 5. DCNN HWA configuration generator tool

A pre-trained network definition and weight data from a development PC are converted into binary DCNN HWA configuration data.

generator tool automatically partitions a network into multiple sub-networks. Network partitioning is performed in two ways: "tile partitioning" that divides intermediate image data of DCNN into tiles in both vertical and horizontal directions and "weight partitioning" that divides weight data into smaller chunks for each output channel. **Figure 6** shows an example of tile partitioning. Figure 6(a) is a network prior to optimization. Suppose that the sum of the sizes of data A required for the execution of operation b and data B of its result exceeds 1 Mibyte. Then, both data (A and B) cannot be held in the internal memory simultaneously (Figure 6(b)). This means that the DCNN HWA cannot perform operation b. To solve this problem, the DCNN HWA configuration generator tool partitions the network to multiple tiles so that these data can fit in the internal memory. In the example of Figure 6(c), the network is horizontally partitioned into two tiles. The DMA unit fetches part of data A (A1) into the internal memory (#1 in Figure 6(d)). Then, the execution unit performs operation b1 on A1 and generates data B1. The DMA unit, in turn, writes B1 into external DRAM (#2 in Figure 6(d)). Likewise, the execution unit performs an operation on data A2 (#3 in Figure 6(d)), and the DMA unit writes its result (B2) into external DRAM (#4 in Figure 6(d)). As a result, the DRAM now holds the same data as B obtained prior to tile partitioning.

(2) Sub-network integration

Each operation generates intermediate data between layers. If a large amount of data is written to external DRAM, power consumption increases by DRAM write accesses. To avoid this problem, the DCNN HWA configuration generator tool combines multiple operations for the sub-networks created by Step (1) above. This optimization makes it possible to perform the next operation without moving intermediate data from the internal memory to external DRAM. In Figure 6(d), data B1, i.e., the result of operation b1, is used as input to operation c1. However, since both b1 and c1 are partitioned into tiles, it is necessary to write data B1 into external DRAM (#2 in Figure 6(d)) and then read it back into the internal memory (#5 in Figure 6(d)). To eliminate this DRAM access, the DCNN HWA configuration generator tool configures a sub-network for b1 and c1 in such a manner that operation c1 uses B1 in the internal memory generated by operation b1 (#3 in Figure 6(e)). This eliminates the need to write and read data B1 and B2 to/from external DRAM, reducing the amount of data transfer due to DRAM access.

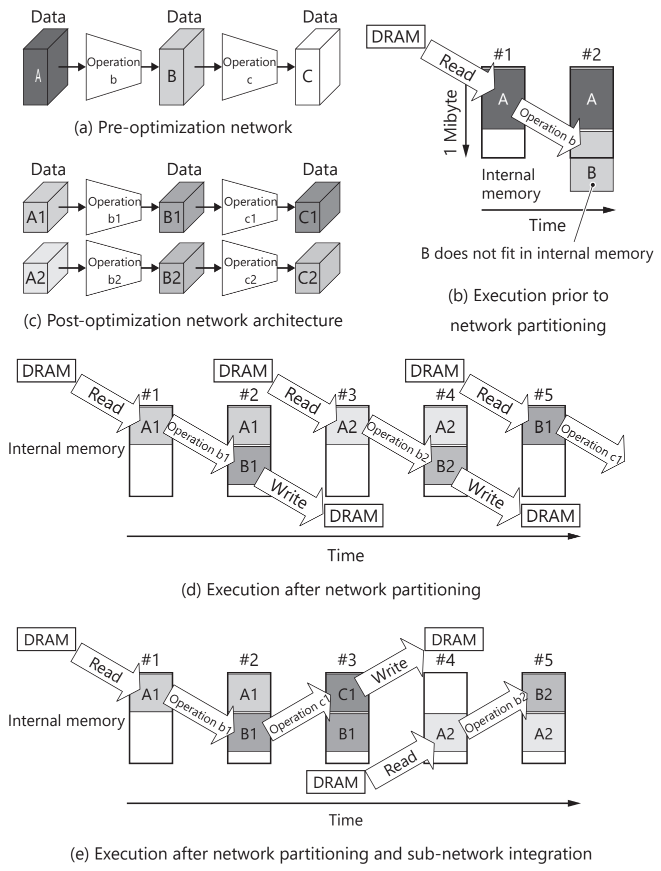
Data
A
Operation b
B
Operation c
C
(a) Pre-optimization network
DRAM
Read
#1
#2
A
Operation b
1 Mibyte
Internal memory
B
Time
B does not fit in internal memory
(b) Execution prior to network partitioning
A1
Operation b1
B1
Operation c1
C1
A2
Operation b2
B2
Operation c2
C2
(c) Post-optimization network architecture
Internal memory
Read
Write
#3
#4
#5
(d) Execution after network partitioning
(e) Execution after network partitioning and sub-network integration

Figure 6. Example of optimization for reduction in amount of data using DCNN HWA configuration generator tool

Network partitioning and sub-network integration make it possible for the DCNN HWA to efficiently execute DCNN processing that requires a huge amount of data.

4. Measurement of power consumption of DCNN processing

We measured the power consumption of the DCNN HWA using a prototype large-scale integration (LSI) chip with the DCNN HWA. As a result, the power consumption of the DCNN HWA was measured to be roughly 1.5 W[3]. **Figure 7** shows an example of semantic segmentation derived from DCNN processing using the prototype chip. The power consumption of the evaluation board was measured to be roughly 6 W (including the power loss of the AC (alternating current) adapter and the power consumption of peripheral interfaces, DRAM chips, and other peripheral devices). The power consumption of a GPU can be of the order of 100 W because of a huge memory bandwidth, an enormous amount of MAC units, and general-purpose versatility. In contrast, the DCNN HWA is specifically designed for DCNN processing. Because of its specialized circuit architecture and efficient memory access, the DCNN HWA performs DCNN processing with low power consumption, eliminating the need for a cooling fan.

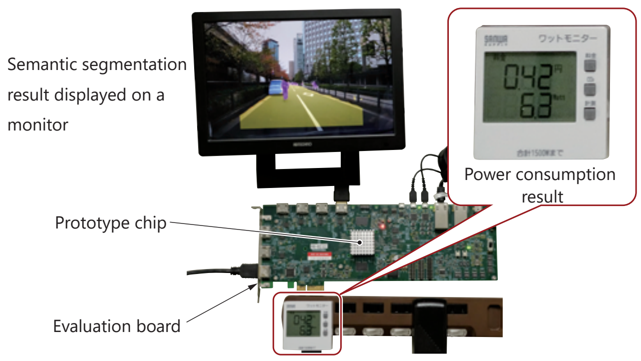

Figure 7. Measurement of power consumption of evaluation board when running semantic segmentation on DCNN HWA
The power consumption of the evaluation board was measured to be roughly 6 W when running image segmentation on a prototype chip.

5. Conclusion

We have developed the DCNN HWA and the DCNN HWA configuration generator tool. The DCNN HWA reduces the number of memory accesses and the amount of memory read and write data by means of parallel DRAM access and computing operations, a specialized circuit architecture, and a dedicated tool. As a result, the DCNN HWA realizes DCNN processing with a power consumption of roughly 1.5 W. An evaluation board with the DCNN HWA achieves power consumption as low as only roughly 6 W, eliminating the need for a cooling fan. We are planning to incorporate the DCNN HWA into Visconti5, our automotive image recognition processor series, to contribute to realization of high-accuracy and sophisticated ADAS using DCNN processing. In addition, we will use the DCNN HWA for various products since there is also strong demand for low-power-consumption DCNN processing for non-automotive applications.

References

(1) Simonyan, K.; Zisserman, A. 2015. "Very Deep Convolutional Networks for Large-Scale Image Recognition." Proceedings of 3rd International Conference on Learning Representations (ICLR 2015). San Diego, CA, 2015-05, ICLR. arXiv.org e-Print archive, 2015, arXiv: 1409.1556v6. Accessed June 5, 2019. https://arxiv.org/pdf/1409.1556.pdf.

(2) Jia Y. et al. 2014. "Caffe: Convolutional Architecture for Fast Feature Embedding." Proceedings of the 22nd ACM International Conference on Multimedia (ACM Multimedia 2014). Orlando, FL, 2014-11, Association for Computing Machinery: 675–678.

(3) Yamada, Y. et al. 2019. "A 20.5TOPS and 217.3GOPS/mm2 Multicore SoC with DNN Accelerator and Image Signal Processor Complying with ISO26262 for Automotive Applications." 2019 IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers. San Francisco, CA, 2019-02, IEEE: 132–134.