DNPU: An Energy-Efficient Deep-Learning Processor with Heterogeneous Multi-Core Architecture

Dongjoo Shin, Jinmook Lee, Jinsu Lee, Juhyoung Lee, and Hoi-Jun Yoo
Korea Advanced Institute of Science and Technology

An energy-efficient deep-learning processor called DNPU is proposed for the embedded processing of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in mobile platforms. DNPU uses a heterogeneous multi-core architecture to maximize energy efficiency in both CNNs and RNNs. In each core, a memory architecture, data paths, and processing elements are optimized depending on the characteristics of each network. Also, a mixed workload division method is proposed to minimize off-chip memory access in CNNs, and a quantization table-based matrix multiplier is proposed to remove duplicated multiplications in RNNs.

Deep learning, formally called deep neural networks (DNNs), has been widely studied and adopted everywhere due to its overwhelmingly high performance and infinite applicability. There are three main types of DNNs. Multilayer perceptron (MLP)[1] is the initial form of neural networks. Convolutional neural networks (CNNs)[2] have convolutional layers with powerful capabilities for extracting visual features, and recurrent neural networks (RNNs)[3] have feedback connections and maintain their internal states. In addition, combining CNNs and RNNs enables visual entity recognitions that vary with time, such as gesture and action recognitions[4] and image captioning.[5]

Due to massive computational requirements of deep-learning algorithms, DNN dedicated processor SoCs[6-10] have been recently introduced to achieve higher energy efficiency. Previous works have considered the acceleration of CNNs[6-8] or MLPs.[9] However, there has been no study of the combined CNN-RNN processor SoC. Only recently, Google announced a custom ASIC processor called a tensor processing unit (TPU)[10] for accelerating MLPs, CNNs, and RNNs. However, the TPU's 40-W power consumption and 28-Mbyte on-chip memory are not suitable for mobile

Published by the IEEE Computer Society
0272-1732/18/$33.00 ©2018 IEEE

devices. Also, due to the following considerations, the TPU's homogeneous architecture is not perfect for mobile low-power implementation. CNNs require a large amount of computation with a relatively small number of filter weights, but MLPs and RNNs require a relatively small amount of computation using a large number of filter weights. Also, their data reusability, required data precision, computational patterns, and memory requirements are very different.

In this article, we propose a CNN-RNN SoC named DNN processing unit (DNPU) to enable embedded DNN processing in mobile devices. It has a heterogeneous architecture to support both CNN and RNN efficiently. Also, an off-chip access-efficient workload division method is proposed to handle large data with a limited on-chip memory and a quantization table-based matrix multiplier to remove duplicated multiplications.

DESIGN CHALLENGES

Heterogeneous Characteristics

The first challenge is heterogeneous characteristics in DNNs. Figure 1 shows the computation and parameter analysis of CNNs, MLPs, and RNNs. While CNNs require a huge amount of MAC operations with a relatively small number of parameters, MLPs and RNNs require a relatively small amount of computations with a large number of parameters.

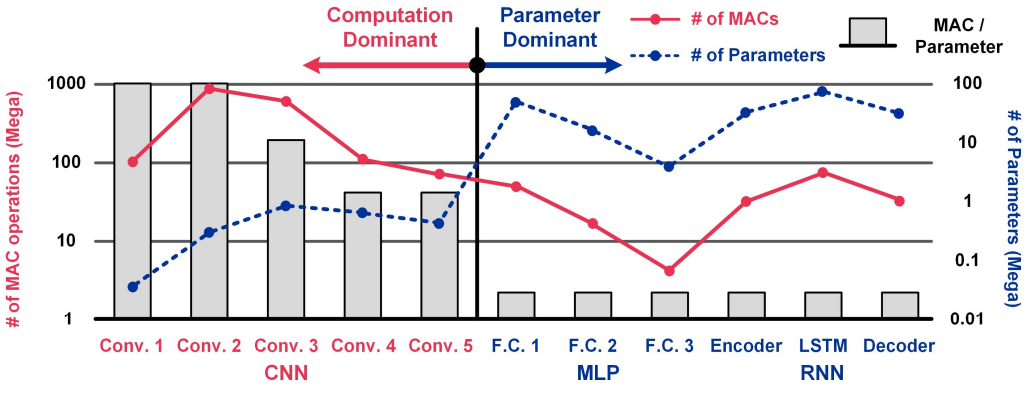


Figure 1. Heterogeneous characteristics of DNNs. CNNs have a computation dominant characteristic, while MLPs and RNNs have a parameter dominant characteristic.

Various Configurations

Even for the same type of network layers, network configurations are very diverse. Particularly, in the case of the convolutional layer, a number of input channels, a number of output channels, an input image size, a filter size, and a stride can all have different values in each layer. In VGG-16 network,[11] the filter size and the stride of each convolution layer are fixed (filter size: 3x3, stride: 1); however, the input image size, input channel number, and output channel number are very different in each layer. In this work, we propose the mixed workload division method to minimize off-chip memory access under various configurations of convolutional layers.

Massive Parameters and Computations

Lastly, deep learning requires a large number of parameters and massive MAC operations. The number of kernel weights and the number of MAC operations vary for layers, but the total number of kernel weights is about 70 million and the total number of MAC operations is 14 billion for only one inference at VGG-16.

PROCESSOR ARCHITECTURE

Overall Architecture

Figure 2(a) shows the heterogeneous architecture of the proposed DNPU. It consists of a CNN processor, an MLP-RNN processor, and a top RISC controller. The CNN processor is designed to maximally exploit the reusability of convolution operations and has a large number of processing elements (PEs) to cover the massive MAC operations. On the other hand, the MLP-RNN processor has a smaller number of processing elements, but it is optimized to reduce the off-chip access for the huge number of parameters. The CNN processor is composed of 16 convolution cores and one aggregation core. Four convolution cores are serially connected, and the last convolution core is connected to the aggregation core. Partial sum results of convolution operation are transferred to the next core and accumulated. The convolution operation is done with 1D form, and three PEs are serially connected to process a 3x1 unit kernel. To process a 3x3 kernel, three 3x1 unit kernels are needed. In the case of a 5x5 kernel, we should process with a 6x5 kernel, so the PE utilization is 5/6. As shown in Figure 2(b), the MLP-RNN processor performs matrix multiplications with eight 16-entry quantization tables (Q-tables), and eight 16b fixed-point multipliers are used to update the Q-tables and do element-wise multiplications. Also, it has activation lookup tables to support tanh and sigmoid functions.

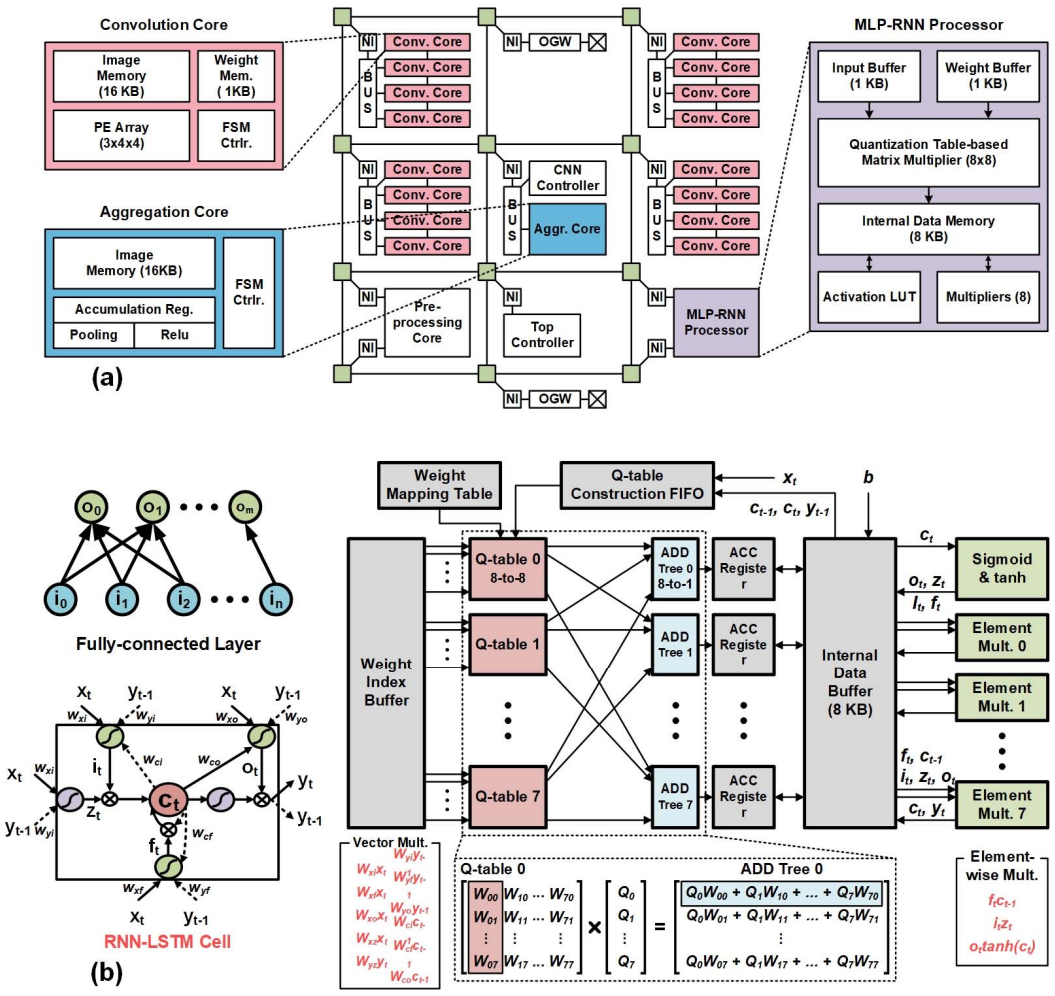


Figure 2. The architecture of DNPU. (a) Overall architecture. DNPU has a heterogeneous architecture with a CNN processor and an MLP-RNN processor. (b) The detailed architecture of the MLP-RNN processor.

Heterogeneous Architecture

DNPU has a heterogeneous architecture with the CNN processor and the MLP-RNN processor separately. A unified architecture for CNNs, MLPs, and RNNs would be more promising, because it can have higher generality and scalability. Also, the unified architecture can take full advantage of core resources across all workloads with CNNs and RNNs (or MLPs). In the heterogeneous architecture, the CNN processor cannot be used in MLP- or RNN-only workload cases and vice versa. Figure 3 shows these results. The unified architecture can achieve 100-percent utilization with every workload. (It is assumed that there is no degradation of PE utilization due to the neural network configurations.) In practice, however, the external memory bandwidth is limited. Therefore, only a certain amount of workload can be computed by the processor. In the case of CNNs, both the feature map and the weight are reused many times. However, in the case of MLPs and RNNs, the weight is used for only one multiplication. Therefore, constraint by the external memory bandwidth is much larger. Therefore, there is almost no throughput loss caused by using the heterogeneous architecture. DNPU has a heterogeneous architecture to maximize energy efficiency. The reason that the heterogeneous architecture can have higher energy efficiency than the unified architecture is that it can be further optimized for each characteristic in the following sections.

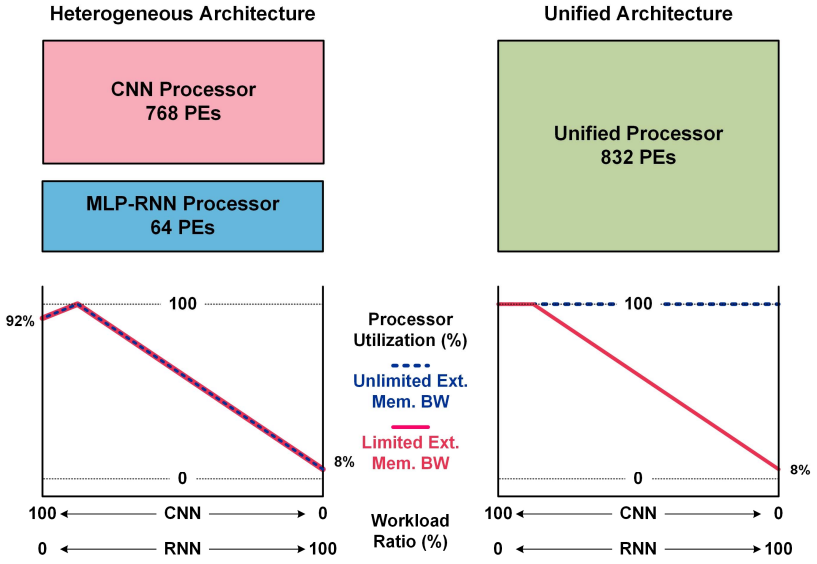


Figure 3. Processor utilization under a heterogeneous architecture and a unified architecture.

Data reuse

There are two main types of reuse patterns in processing at convolutional layers: One is input image reuse and the other is kernel weight reuse. The input image is reused from several tens to hundreds of times, while the kernel weights are reused from hundreds to several tens of thousands of times. In the case of MLPs or RNNs, input values have reuse characteristics, but weights are not reused with different values.

ALU

To achieve high energy efficiency, DNN processing using reduced precision has been extensively studied and widely used. Simulation results show that MLPs and RNNs do not fall in accuracy even when weights are quantized to 16 levels with k-means clustering. The MAC unit can also be optimized for the reduced precision and required data path. For the MLP-RNN processor, we propose the quantization table-based matrix multiplication to remove the duplicated multiplications.

On-chip memory

To process DNNs, we basically need on-chip memory for input feature map, output feature map, and weights. Because of the limited size of on-chip memory, off-chip memory access varies greatly depending on how on-chip memory is used. For CNN processors, input feature memory is dominant with input parallel processing. In the input parallel processing, convolution results from all input channel images that are accumulated to generate one output channel image at a time. To operate in this way, on-chip memory corresponding to CH_{in} input images and one output image is required. In the case of MLPs and RNNs, there is no reuse opportunity for the weights, however, input features can be reused as many times as the number of output features. Therefore, it is advantageous to use an output parallel approach, and output feature memory is dominant for the MLP-RNN processor.

ENERGY-EFFICIENT FEATURES

Mixed Workload Division Method

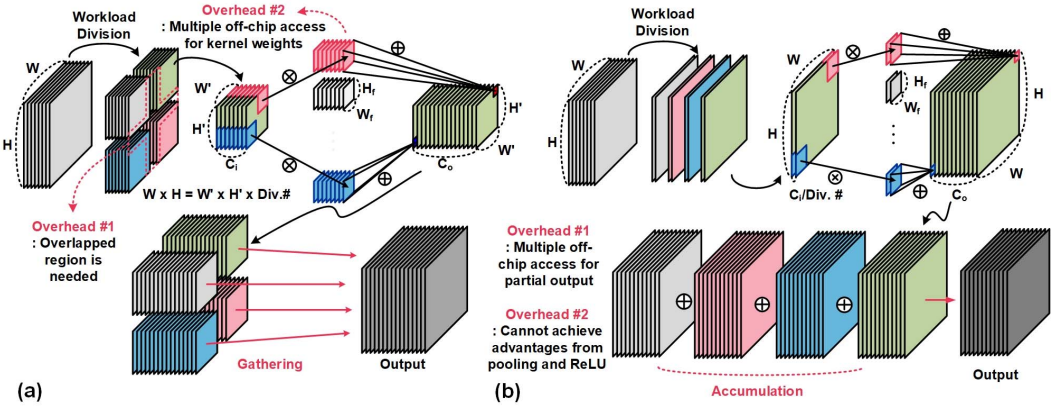
The mixed workload division method is proposed to handle various configurations in convolutional layers. In convolution operations, input layer images are multiplied with convolution kernels to generate output images. Each of the input layer and output layer images, and convolution kernel weights, is about 10 to 100 Mbytes. However, an on-chip memory size is limited to a few hundred Kbytes or a few Mbytes. Therefore, workloads should be divided to reduce the required on-chip memory size.

There are three possible division methods: image, channel, and mixed. In the case of image division (see Figure 4(a)), width and height of the input images have values reduced in proportion to the division number. Each of the divided image groups is processed independently. In this case, kernel weights should be fetched multiple times for each divided image group. Therefore, memory access for the weight is increased. On the other hand, in channel division (see Figure 4(b)), the width and height of the input images are the same as the width and height of the original input images. Alternatively, it has multiple reduced channels. In this case, the fetched kernel weights can be used for whole images, so it needs no multiple off-chip access. However, in this case, partial output results should be accumulated to generate the final output, which requires additional off-chip memory access. These results are summarized in Figure 4(c). Image division has an advantage when the weight size is relatively smaller than the image size. On the other hand, channel division has an advantage when the image size is relatively smaller than the weight size. These two methods combine in mixed division, which uses both image and channel division. In the VGG-16 example, in the front layers, image size is large, but the channel size is small. And if the layer progresses, the image size gets smaller and the channel size gets bigger. In this environment, the effect of mixed division is shown in Figure 5(a) and Figure 5(b).

Quantization Table-based Multiplier

In MLPs and RNNs, weight can be quantized (meaning clustered) to the smaller number of levels. If original weights have a 16-bit fixed-point data type, weights can have 2^{16} different values. However, if the weights are quantized with a 4-bit index, weights can have $2^4 = 16$ different values. It does not mean that the weight value itself is 4-bit. Figure 6(a) shows simulation results with weight quantization (meaning weight clustering). Four-bit weight quantization shows the negligible error increasing. By using this weight quantization, multiplication can be replaced by table lookup. Figure 6(b) shows the operation of quantization tables. Multiplications between input and quantized weights are also quantized to 16 values. So, each entry of the quantization table contains the pre-computed multiplication result between a 16-bit fixed-point input and a 16-bit fixed-point weight. After the Q-table is constructed once, only quantized indexes are decoded to fetch the pre-calculated entry to do the multiplications. Because we only need to load 4-bit index instead of 16-bit weight, 75 percent of off-chip memory access can be reduced. Figure 6(c) shows the area, power, and latency comparison between a 16-bit fixed-point multiplier and the Q-table-based multiplier. The power consumption is reduced by 79 percent, and the latency

is reduced by 93 percent. The Q-table is large in area because it needs lookup registers. But, one Q-table has eight index inputs and eight multiplication result outputs, so the effective area for one multiplication is less than the area of one 16-bit fixed-point multiplier.



(c)

	Off-chip Access (W/O Compression Scheme)					
	Image Division		Channel Division		Mixed Division	
Input Image	$W_i \times H_i \times C_i$		$W_i \times H_i \times C_i$		$W_i \times H_i \times C_i$	
Output Image	$\frac{W_o \times H_o \times C_o}{\text{Pooling Size}}$		$W_o \times H_o \times C_o \times (\text{Ch. Div. \#} \times 2)$		$W_o \times H_o \times C_o \times (\text{Ch. Div. \#} \times 2)$	
Weight	$W_f \times H_f \times C_i \times C_o \times \text{Img. Div. \#}$		$W_f \times H_f \times C_i \times C_o$		$W_f \times H_f \times C_i \times C_o \times \text{Img. Div. \#}$	

Figure 4. Workload division methods. (a) Image division. (b) Channel division. (c) Off-chip memory access analysis.

Img. Div. #	16	8	4	2	1
Ch. Div. #	1	2	4	8	16
Input Image	1.5	1.5	1.5	1.5	1.5
Output Image	1.5	6.1	12.3	24.5	49
Weight	18	9.0	4.5	2.3	1.1
Total (MB)	21.0	16.6	18.3	28.3	51.6

VGG-16 Example – 6th Convolution Layer

(a) $(W_i, H_i, C_i) = (56, 56, 256), (W_o, H_o, C_o) = (56, 56, 256), (W_f, H_f) = (3, 3)$

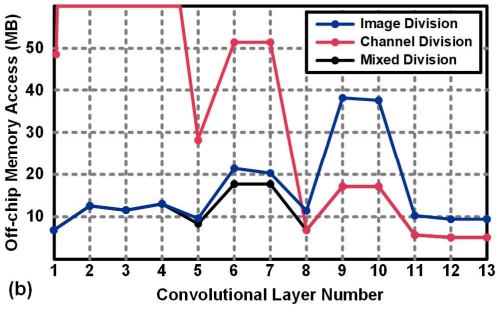


Figure 5. Off-chip memory access analysis on VGG-16. (a) Mixed workload division analysis example on VGG-16 sixth convolution layer. Off-chip memory access is minimized at Image Division #8 and Channel Division #2. (b) Off-chip memory access comparisons.

IMPLEMENTATION

Figure 7 shows the chip photograph and the performance summary of DNPU[12] fabricated using 65-nm, one-poly eight-metal CMOS technology. DNPU occupies a 16-mm^2 die area with 280-Kbyte on-chip SRAM for the CNN processor and 10 Kbytes for the MLP-RNN processor. The processor can operate from a 0.765- to 1.1-V supply with a 50- to 200-MHz clock frequency. The power consumption levels at 0.765 V and 1.1 V are 34.6 mW and 279 mW, respectively. It shows 8.1 TOPS/W energy efficiency with 4-bit word length at 0.77 V.

Table 1 shows a performance comparison with the five previous deep-learning SoCs, and Table 2 shows the measured performance breakdown of the VGG-16 network and a three-layer RNN network with one long short-term memory (LSTM) layer.

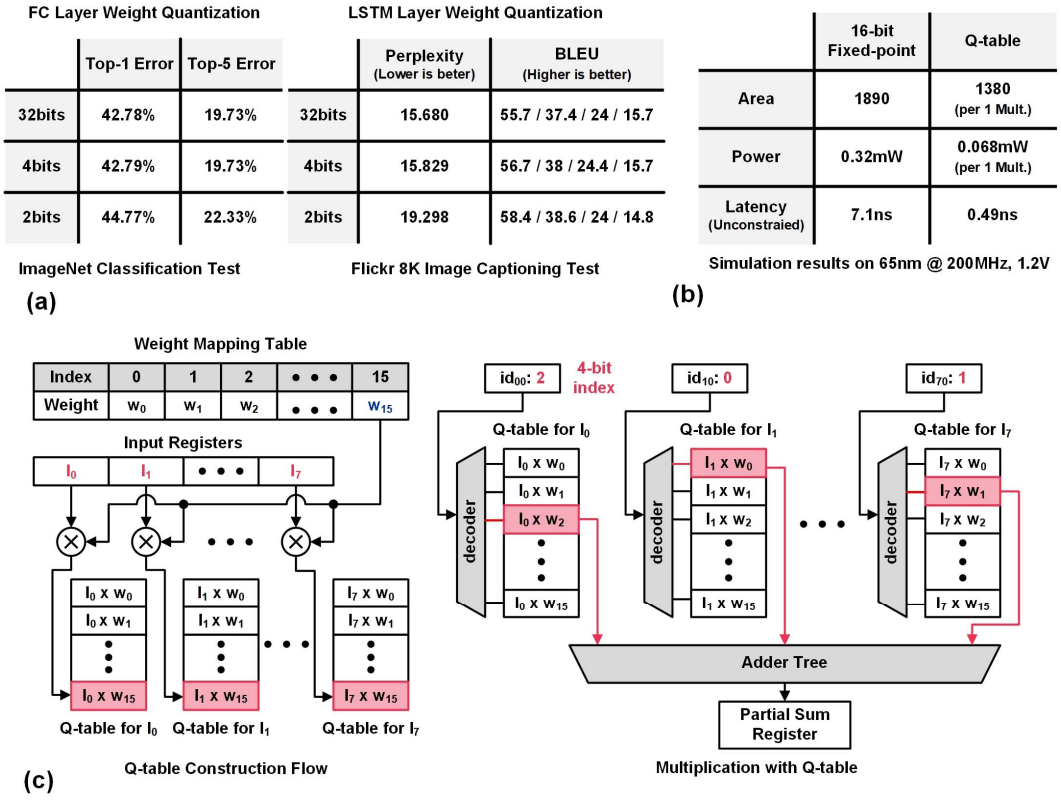
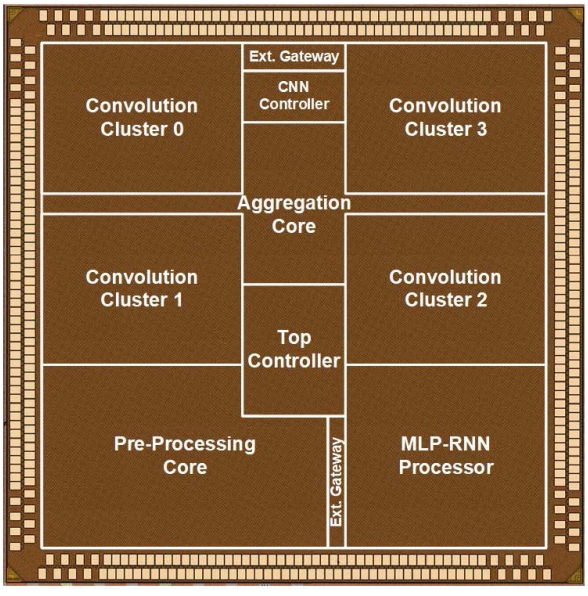


Figure 6. Q-table-based multiplier. (a) Simulation results with weight quantization (weight clustering). (b) Q-table construction flow and multiplication with Q-table. (c) Performance comparison with conventional multiplier.



Parameters	Specifications			
Technology	65-nm 1P8M CMOS			
Die Area	4 mm x 4 mm (16 mm^2)			
SRAM Size	CLP	MLP-RNN		
	280 KB	10 KB		
Supply Voltage	0.77 to 1.1V			
Clock Frequency	CNN	MLP-RNN	NoC	ETC
	~200 MHz	~200 MHz	~400 MHz	~200 MHz
Power		CNN	MLP-RNN	ETC	Total
	50 MHz @ 0.77V	29 mW	2.6 mW	3 mW	35 mW
	200 MHz @ 1.1V	235 mW	21mW	23 mW	279 mW
Energy Efficiency		CNN Word Length: 16-bit		CNN Word Length: 4-bit	
	50 MHz @ 0.77V	2.1 TOPS/W		8.1 TOPS/W	
	200 MHz @ 1.1V	1.0 TOPS/W		3.9 TOPS/W	

Figure 7. DNPU chip photograph and performance summary. The processor is fabricated using 65-nm CMOS technology and occupies 16 mm^2.

Table 1. Performance comparison with previous DNN SoCs.

	Tech. (nm)	Clock Freq. (MHz)	Supply Voltage (V)	CNN	MLP, RNN	Peak Perform. (GOPS)	Power (mW)	Energy Efficiency (TOPS/W)
SoVC 2016 [6]	40	12 – 204	0.55 – 1.1	o	x	102	-	2.6
							@ 12 MHz, 4-bit	
ISSCC 2017 [7]	28	~ 200	0.65 – 1.1	o	x	76	7.6	10
							@ 50 MHz, 4-bit	
ISSCC 2017 [8]	28	200 – 1175	0.575 – 1.1	o	x	676	39	2.93
							@ 200 MHz, 8-bit	
ISCA 2016 [9]	28	1200	-	o	o	400	2360	0.17
							@ 1200 MHz, 4-bit	
ISCA 2017 [10]	28	700	-	o	o	92000	40000	2.3
							@ 700 MHz, 8-bit	
Proposed	65	50 – 200	0.77 – 1.1	o	o	1214	34.6	8.1
							@ 50 MHz, 4-bit	

Table 2. Performance analysis on VGG-16 and RNN.

	Convolutional Layer									MLP			RNN		
	1	2	3	4	...	10	11	12	13	1	2	3	Input	LSTM	Out.
# of Input Channels	3	64	64	128		512	512	512	512	25,088	4096	4096	1000	512 LSTM cells	512
# of Output Channels	64	64	128	128		512	512	512	512	4096	4096	1000	512		2538
Image Height & Width	224	224	112	112		28	14	14	14	1	1	1	1	1	1
# of Filter Weights (M)	0.002	0.04	0.07	0.14		2.2	2.2	2.2	2.2	98	16	4	0.5	2	1.2
# of MAC (Gops)	0.081	1.7	0.86	1.7	...	1.7	0.43	0.43	0.43	0.096	0.016	0.004	0.0005	0.002	0.0012
Throughput (Inference/s)	1022	78	165	80		79	320	320	313	16	98	390	3080	704	1239
Power (mW)	22	31	33	32		31	31	31	31	2.9	2.9	2.9	3.0	2.9	3.0
Efficiency (TOPS/W)	7.5	8.7	8.6	8.6		8.8	8.9	8.9	8.7	1.0	1.1	1.0	1.0	0.9	1.0

*Operating Conditions: Convolution Processor (0.77V, 50MHz), MLP-RNN Processor (0.77V, 25MHz)

CONCLUSION

We propose DNPU, the heterogeneous multi-core CNN-RNN processor SoC, to realize embedded DNN processing in mobile devices. To maximize energy efficiency, the proposed DNPU has a heterogeneous and highly dedicated architecture with energy-efficient features.

REFERENCES

1. S.K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, September 1992, pp. 683–697; https://ieeexplore.ieee.org/document/159058/#full-text-section.
2. Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, MIT Press, 1995.
3. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, MIT Press, 1997.

4. J. Donahue et al., "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," *The IEEE Conference on Computer Vision and Pattern Recognition*, 2015; https://ieeexplore.ieee.org/document/7298878/.
5. O. Vinyals et al., "Show and Tell: A neural image caption generator," *IEEE Conference on Computer Vision and Pattern Recognition*, 2015; https://ieeexplore.ieee.org/document/7298935/.
6. B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," *IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, 2016; https://ieeexplore.ieee.org/document/7573525/.
7. B. Moons et al., "Envision: A 0.26-to-10 TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," *IEEE International Solid-State Circuits Conference*, 2017; https://ieeexplore.ieee.org/document/7870353/.
8. G. Desoli et al., "A 2.9 TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017; https://ieeexplore.ieee.org/document/7870349/.
9. S. Han et al., "EIE: efficient inference engine on compressed deep neural network," *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016; https://ieeexplore.ieee.org/document/7551397/.
10. N. P. Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit," *ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017; https://ieeexplore.ieee.org/document/8192463/.
11. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations*, 2015; https://arxiv.org/abs/1409.1556.
12. D. Shin et al., "DNPU: An 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017; https://ieeexplore.ieee.org/document/7870350/.

ABOUT THE AUTHORS

Dongjoo Shin is a PhD candidate at the Korea Advanced Institute of Science and Technology (KAIST). His research interests include energy-efficient hardware accelerators and multi-core architectures, especially in machine learning and computer vision fields. Shin has a master's degree in electrical engineering from the KAIST. He is a student member of the IEEE. Contact him at imdjdj@kaist.ac.kr.

Jinmook Lee is a PhD candidate at the KAIST. His research interests include energy-efficient DNN accelerator architecture and ASIC implementation, especially with hardware-software co-optimized design. Lee has a master's degree in electrical engineering from the KAIST. He is a student member of the IEEE. Contact him at jinmooklee@kaist.ac.kr.

Jinsu Lee is a PhD student at the KAIST. His current research interests include low-power digital processors and low-power deep learning and intelligent vision SoC with memory architecture and low-power circuit. Lee has a master's degree in electrical engineering from the KAIST. He is a student member of the IEEE. Contact him at jinsulee@kaist.ac.kr.

Juhyoung Lee is a master's student at the KAIST. His current research interests include energy-efficient multi-core architectures and systems, especially focused on AI and computer vision fields. Lee has a bachelor's degree in electrical engineering from the KAIST. He is a student member of the IEEE. Contact him at juhyoung@kaist.ac.kr.

Hoi-Jun Yoo is a professor in the School of Electrical Engineering at the KAIST. He is also the general chair of the Korean Institute of Next-Generation Computing. His research interests include computer vision SoC, body area networks, and biomedical devices and circuits. Yoo has a PhD in electrical engineering from the KAIST. He is a coauthor of several books, including "Embedded Systems" (Wiley, 2012) and "Ultra-Low-Power Short Range Radios" (Springer, 2015). He has received numerous awards, including the Electronic Industrial Association of Korea Award and the Korea Semiconductor Industry Association Award. He is a Fellow of the IEEE. Contact him at hjyoo@kaist.ac.kr.