IEEE Open Journal of
Circuits and Systems

Received 2 October 2021; revised 22 November 2021; accepted 15 December 2021. Date of publication 21 December 2021;
date of current version 19 January 2022.

Digital Object Identifier 10.1109/OJCAS.2021.3137263

A 0.61-μJ/Frame Pipelined Wired-logic DNN Processor in 16-nm FPGA Using Convolutional Non-Linear Neural Network

ATSUTAKE KOSUGE (Member, IEEE), YAO-CHUNG HSU, MOTOTSUGU HAMADA (Member, IEEE), AND TADAHIRO KURODA (Fellow, IEEE)

Graduate School of Engineering, University of Tokyo, Tokyo 113-8656, Japan

This article was recommended by Associate Editor M. Rastogi.

CORRESPONDING AUTHOR: A. KOSUGE (e-mail: kosuge@dlab.t.u-tokyo.ac.jp)

ABSTRACT A pipelined wired-logic deep neural network (DNN) processor implemented in a 16-nm field-programmable gate array (FPGA) is presented. The latency and power required for memory access are minimized by utilizing the wired-logic architecture, thus enabling low power and high throughput operation. One technical issue with the wired-logic architecture is that it requires a lot of hardware resources. To reduce them, two core technologies are developed: (1) a convolutional non-linear neural network (CNNN) and (2) a pipeline-type neuron cell. The CNNN optimizes both the network structure and the non-linear activation function of each neuron by using a newly developed back-propagation-based training method. While conventional reinforcement learning can train only a small size network thus limiting its application to handwritten number recognition, the proposed CNNN enables a larger network size making it applicable to object recognition. The pipeline-type neuron cell has a small look-up table (LUT) to process non-linear functions using only a small amount of hardware resources. These two technologies enable the implementation of the entire network on a single FPGA chip with the wired-logic architecture. Three types of CNNN trained on the CIFAR-10 dataset are implemented in 16-nm FPGAs. An energy efficiency of 0.09, 0.12, and 0.61 μJ/frame is achieved with 70%, 75%, and 82% accuracy, respectively. Compared with a state-of-the-art accelerator using a binary neural network (BNN), the energy efficiency is improved by more than two orders of magnitude.

INDEX TERMS Edge computing, FPGA, on-device intelligence, deep learning, software and hardware co-design.

I. INTRODUCTION

THE DEEP neural network (DNN) is a promising technology for IoT solutions. The incorporation of a DNN, especially a convolutional neural network (CNN), into an edge system is expected to expand its application into areas such as the automation of human workload in warehouses [1] and factories, smart cities, and unstaffed retail shops. Always-on AI cameras [2] are a promising solution to realize such IoT applications.

The technical challenge in applying DNN to edge systems is energy efficiency. The weight parameters of synapses and data are transferred from memory to arithmetic unit circuits to execute a large amount of multiply-accumulate (MAC) operations. Since there are a lot of parameters in DNNs (Fig. 1(a)), the memory access needs to be performed many times for each DNN calculation (Fig. 1(b)). A large amount of power is consumed in memory access, especially in DRAM access which consumes two orders of magnitude more power than the arithmetic unit [3]. Since most IoT systems have a limited battery capacity, it is necessary to make power consumption as low as possible.

To improve the energy efficiency in DNN processing, bit-width reduction techniques have been actively studied [4]–[7]. Since DNNs have a lot of weight parameters, high recognition accuracy can be obtained even with a reduced bit width [7]. By reducing the bit width, the total

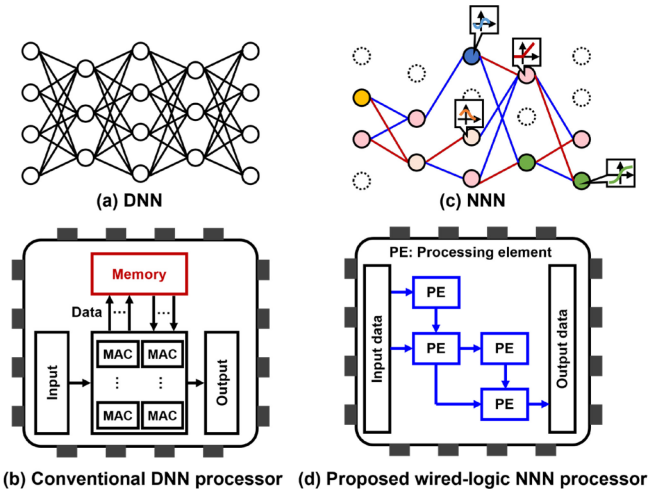


FIGURE 1. Comparison of DNN accelerator architectures. (a) Conventional DNN and (b) its processor.(c) Proposed NNN and (d) its wired-logic processor.

amount of data processed can be reduced to fit into on-chip SRAM which has limited storage capacity. Since the power consumption for SRAM access is lower than that for DRAM [3], the total power consumption can be lowered.

Recently, the bit width of data and weight parameters has been aggressively reduced to a binary bit, resulting in what is known as a binary neural network (BNN). By using BNN, the processor can process the neural network with on-chip SRAM only to improve energy efficiency [8]–[13]. The most energy efficient BNN processor achieves an energy efficiency of 3.8 μJ/image [9]. However, it still consumes power that is at least 5.7 times larger than the power budget for the AI-based always-on smart camera application [2].

One promising solution to improve energy efficiency is the wired-logic architecture where all the processing elements (PEs) required for DNN processing are implemented on one chip and data are transferred from PE to PE directly. Intermediate data are never stored in either SRAM or DRAM. Unlike the conventional processor architecture, which allows flexible processing according to input instructions, the wired-logic architecture can only repeat fixed processing. However, the wired-logic architecture is quite energy-efficient and its latency is quite small due to the elimination of memory access. The wired-logic architecture is typically used in the control field, where only fixed processing is performed, and latency is the most important factor [14]. It can also be applied to DNN because the same trained DNN model is processed iteratively in AI-based IoT systems such as robots in warehouses [1].

The technical problem with applying the wired-logic architecture to DNN lies in the required hardware resources. Specifically, the conventional DNN model requires a lot of computing elements (neurons and synapses), thereby consuming a huge amount of hardware.

To mitigate this problem, a non-linear neural network (NNN) (Fig. 1(c)) that can achieve the same recognition accuracy with fewer neurons and synapses, along with its wired-logic processor implementation in FPGA (Fig. 1(d)), was developed [15]. Both the non-linear activation functions of neurons and the network structure of the neural network were optimized by using a reinforcement learning technique. The number of neurons was reduced by more than one order of magnitude and the number of synapses was reduced by more than two orders of magnitude while keeping the same recognition accuracy. The wired-logic FPGA-based DNN processor [15] is 47 times more energy-efficient than a conventional SRAM-based digital ASIC BNN processor [8].

One technical issue with this conventional NNN is that the network size is limited. Since the search space of reinforcement learning is limited to small-scale problems (e.g., using only a few thousand neurons), the application of the NNN is also limited to the handwritten number recognition of black-and-white images. It is difficult to apply it to object recognition tasks with color images because a much more complex neural network structure with a larger number of neurons would be required.

In this work, a convolutional NNN (CNNN) is proposed that can be applied to a broader range of applications. Wired-logic processors using the proposed CNNN are demonstrated with the CIFAR-10 [16] color-image object recognition dataset. The following two core technologies are developed.

1) Convolutional NNN (CNNN): Both the network structure of the NNN and the non-linear activation function of each neuron are optimized by a newly developed back-propagation-based training method. A conventional CNN is used as the initial network structure to enable stable and fast learning even with complex tasks such as object recognition.

2) Pipelined Neuron Cell: In CNNN, all the weights are binarized (+1 or −1), so multiplication is not necessary and only adders and subtractors are required. Instead, calculating a different nonlinear function for each neuron is required. To calculate the nonlinear functions efficiently, an activation look-up table (Act-LUT) technique is used. A pre-calculated activation function is stored in an LUT that outputs the value of the function in accordance with the input. Act-LUTs are implemented with FPGA LUTs. Since the developed CNNN inherits the CNN structure, there are no wires straddling layers. Therefore, it can easily be implemented in a pipelined architecture to increase throughput.

Using the above technologies, CNNNs trained on the CIFAR-10 dataset were implemented in 16-nm FPGAs. Three types of CNNNs (Small/Middle/Large) were realized which demonstrated a recognition accuracy of 70.6%, 75.3%, and 81.6%, respectively. Compared with the state-of-the-art BNN processor implemented on an FPGA [10], which achieves an energy efficiency of 164 μJ/frame, the energy efficiency is improved by more than two orders of magnitude (at 0.61 μJ/frame). By making the proposed wired-logic processor an ASIC chip and reducing static power consumption, the power consumption can be lowered further to sub-mW level, making it suitable for always-on AI cameras [2].

TABLE 1. Comparison of conventional neural network architectures.

	DNN	Neuromorphic Ref. [17, 22]	Conv. NNN Ref. [15]	Proposed CNNN
Learning method	Back-propagation	STDP	Biological reinforcement	Back-propagation
Learning parameters	Synapse weight	Spike timing	1) Synapse connection 2) Activation	1) Synapse binary weight 2) Activation
Number of synapse, neuron	1x	1x	< 0.1x	< 0.1x
Processor type	µProcessor	Wired-logic processor		
	Applicable image recognition datasets			
MNIST	✓	✓	✓	✓
CIFAR-10	✓	✓	-	✓
ImageNet	✓	-	-	-

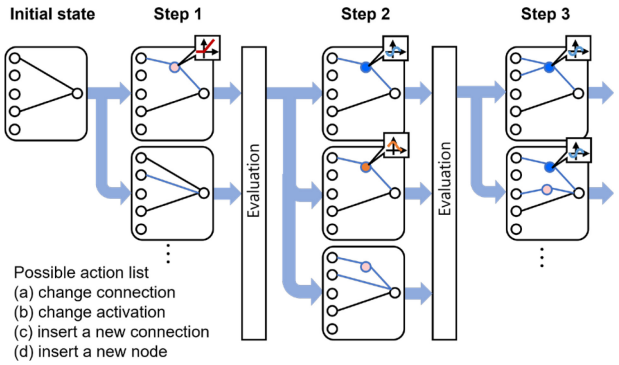


FIGURE 2. Conventional reinforcement learning algorithm for NNN.

Section II of this paper gives an overview of related research on wired-logic DNN accelerators. In Section III, we present the proposed CNNN method after describing the conventional NNN learning method based on reinforcement learning and its problems. In Section IV, the proposed wired-logic accelerator and its implementation are described. The experimental results are reported in Section V. This paper concludes with a brief summary in Section VI.

II. RELATED WORKS ON WIRED-LOGIC DNN PROCESSORS

A well-known method for implementing neural networks in a wired-logic architecture is a neuromorphic processor [17]–[22]. Neurons and synapses that mimic the human cerebrum are integrated onto silicon chips with an algorithm called a spiking neural network (SNN) that mimics neural activities. Similar to the human cerebrum, data signals are transferred directly between neurons via synapses, without being stored in memory. To further improve energy efficiency, analog-type neuron circuits have been actively studied [20], [21].

The problem with the neuromorphic processor and SNN algorithm is a large chip size. While the SNN mimics neural activities, expressive ability of the neural network is low, resulting in a large number of neurons and synapses (Table 1). Almost the same numbers of neurons and synapses as conventional DNN are required [17]. To implement such a large SNN model, 8 silicon chips are needed, resulting in a large power consumption [17].

In an attempt to minimize the number of neurons and synapses, thereby achieving an area-efficient wired-logic DNN processor, a non-linear neural network (NNN) has been developed [23]. In this NNN, both the neural network structure and the non-linear function of each neuron are optimized by reinforcement learning. By properly using neurons with a wide variety of non-linear functions, the expressive ability of the neural network is improved. Therefore, high recognition accuracy can be achieved with a small number of neurons and synapses. Another proposed network is a fixed-weight NNN (FW-NNN) in which all weights are fixed to a constant value such as "+1" [15]. By using FW-NNN, multiplication of data and weights becomes unnecessary, resulting in higher energy efficiency.

The problem with NNN is that training is difficult with complex tasks such as object recognition tasks due to the evolutionary reinforcement learning algorithm. With complex tasks, the search space becomes quite large, so it takes a long time to find an optimal solution. Similar to the neuromorphic method, the application of NNN has thus far been limited to MNIST tasks [15], [23].

III. LEARNING METHOD OF NON-LINEAR NEURAL NET

A. CONVENTIONAL REINFORCEMENT LEARNING

Conventional training of NNN [15], [23] is done through reinforcement learning using a NeuroEvolution of Augmenting Topologies (NEAT) algorithm [24]. NEAT is a genetic algorithm to generate neural networks by using genetic operations such as selection, mating, and mutation. The detailed NNN training procedure is as follows. First, various actions from the following candidates are randomly selected at each step: (a) change connection, (b) change activation, (c) insert a new connection, and (d) insert a new node (Fig. 2). Then, the neural network structures generated at each step are evaluated, and those with low scores are eliminated. The evaluation criteria are recognition accuracy and the number of neurons. These steps are repeated until a final neural network is generated.

The key feature of the conventional NNN is that the expressive ability of the neural network can be improved by using an individually optimized nonlinear activation function in each neuron. For example, non-linear functions such as Sigmoid, Relu, Tanh, Sin, Cos, and Gaussian which are not used in traditional DNNs are used in the conventional NNN. High handwritten number recognition accuracy is achieved with a small number of neurons even when all synaptic weights are fixed at +1 [15].

The technical challenge here is that the learning is not well processed with complex tasks such as object recognition. The learning of the neural network structure is a kind of combinatorial optimization problem such as determining how to combine N neurons to obtain high recognition accuracy. As the number of neurons N increases, the time complexity

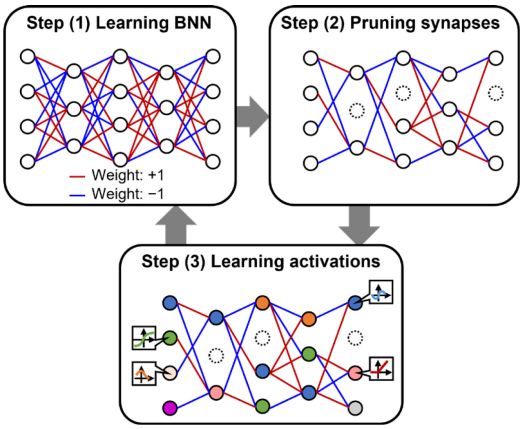


FIGURE 3. Proposed convolutional NNN training algorithm.

increases exponentially ($O(2^N)$). For complex tasks such as object recognition that uses color images as input, such as CIFAR-10, the number of neurons N of the traditional DNN is more than 100 times that for MNIST. The training of MNIST [15] was completed in about two weeks, but the accuracy of the network for CIFAR-10 did not change from the initial value even after a month of training. No network with good accuracy could be obtained in a realistic amount of time with such reinforcement learning technique.

As such, a new learning method is required to expand the application range of NNN. A Graph-CNN-based reinforcement learning method was developed for the combinatorial optimization problem with large N in [25], but it was optimized for the place-and-route problem in VLSI design and not for the training of neural networks. In recent years, network architecture search (NAS) methods have been extensively studied in the field of machine learning, including image recognition tasks [32]–[35]. NAS explores the entire network architecture [32], [33] and the component cells [34], [35]. However, all of them search for combinations of multiple layers, and do not optimize the connections of each neuron as in NNN. The NAS method is difficult to be used for the NNN method as is.

B. PROPOSED CONVOLUTIONAL NNN (CNNN)

In this work, a new NNN training method based on an existing CNN structure is developed. Starting with the CNN, which is already known to provide good recognition performance with CIFAR-10, the initial CNN is transformed into a CNNN as the training proceeds. As with conventional NNN, the weight coefficients are fixed at $+1/-1$. By setting the weighting coefficients to $+1/-1$, multiplication can be eliminated as in the conventional NNN, resulting in improving the efficiency of both the power and area of the arithmetic unit. As shown in [15], the power and area per neuron cell can be reduced by a factor of four in FPGA by removing multipliers. To further reduce power consumption by implementing NNN as an ASIC chip, the binary weight is advantageous because it does not require multiplication and the memory which stores weight coefficients. The detailed procedure is as follows (Fig. 3).

Step (1): Learn the binary weights ($+1/-1$) of the CNN.
Step (2): Prune the synapses of the CNN.
Step (3): Learn the non-linear activation function at each neuron individually.
Step (4): Repeat Steps (1)–(3) until enough accuracy is achieved with the targeted number of neurons.

As described later, **Steps (1) and (2)** are the same as for the conventional CNN using a back-propagation algorithm. Thus, the training proceeds easily even with a large network model. By binarizing the weights ($+1/-1$), the proposed CNNN can be calculated simply by adding and subtracting, without multiplication, the same as the conventional NNN [15]. By using the proposed method, the generated CNNN inherits some of the features from the CNN (e.g., convolution and pooling), unlike the conventional NNN.

The key distinction of the proposed CNNN lies in **Step (3)**, where the activation function of each neuron is optimized by learning. Compared with CNN, where the nonlinear functions of all neurons are Relu, the expressive ability is improved by optimizing the nonlinear functions individually.

Unlike the conventional NNN (Section III-A), where the optimal nonlinear functions are searched by reinforcement learning, the nonlinear functions whose non-linearity is controlled by trainable parameters are optimized by the back-propagation method, as described later. Since the training can be processed with back-propagation just like a conventional CNN, it is possible to achieve a larger CNNN model that can infer with high accuracy even for complex datasets and tasks.

Techniques for both **Step (1)** (training the BNN) and **Step (2)** (pruning the synapses) to reduce the total amount of CNN computations have been studied extensively. The lottery ticket hypothesis method is attracting attention because it enables simple back-propagation-based learning rules while achieving high pruning rates [26].

The multi-prize lottery ticket hypothesis (MLT) [27] is a method that combines BNN with the lottery ticket hypothesis (synapse pruning). It is known that even if BNN is pruned by 80%, training with the MLT method makes it possible to achieve a recognition accuracy equivalent to that of the original, full precision bit width, dense neural network. In the MLT method, a value called the pruning score is set for each synapse according to the initial weight coefficient. The pruning score is an indicator of the importance of each synapse. The indicator is updated by the backpropagation method. After the training is completed, synapses with a pruning score below a predetermined threshold are removed. This synapse pruning operation generates neurons that have no synaptic connections at all, and these are pruned as well.

In this study, the MLT method is utilized in both **Steps (1) and (2)**.

In **Step (3)**, to train the nonlinear function, the parameters (p_1, p_2, p_3) of the following Eq. (1) are optimized [28].

$$y = (p_1 - p_2)/(1 + \exp(-p_3(p_1 - p_2)x) + p_2 x. \quad (1)$$

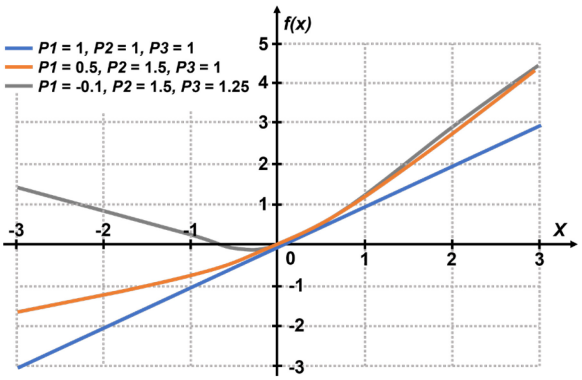


FIGURE 4. Examples of parametrized non-linear function (Eq. (1)).

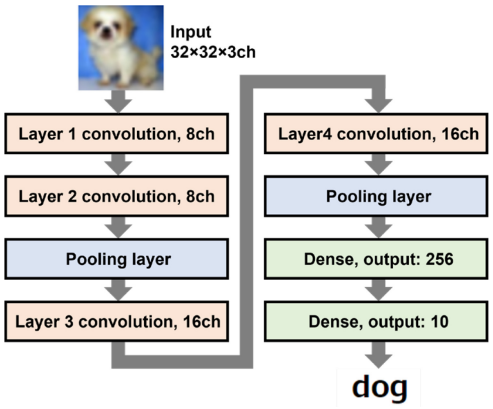


FIGURE 5. Baseline convolutional neural network.

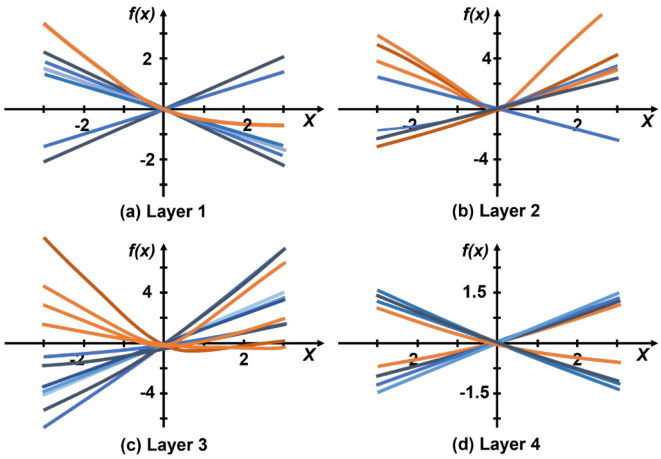


FIGURE 6. Trained results of non-linear functions.

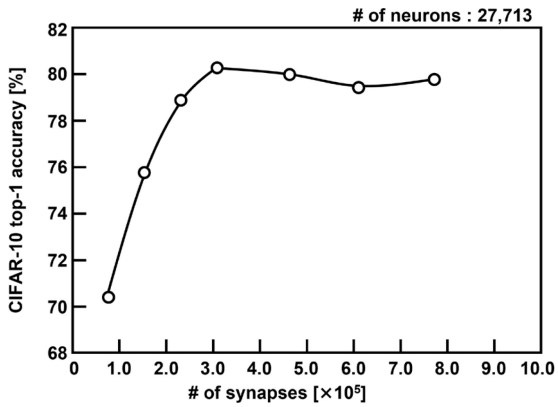


FIGURE 7. Recognition accuracy dependency on number of synapses.

Various nonlinear functions can be achieved by using these parameters (p_1, p_2, p_3). They are optimized at each neuron by the error back-propagation method. Example shapes of the nonlinear function are shown in Fig. 4. It can simulate a wide variety of functions such as the linear function, a Relu-like function, and a downwardly convex function.

In **Step (4)**, **Steps (1)-(3)** are repeated until enough accuracy is achieved with the targeted number of neurons. Training of the BNN and pruning are processed again by using MLT (**Steps (1), (2)**). Each neuron has its own nonlinear function trained in **Step (3)**. In all steps, the training is processed with the back-propagation method, which makes it possible to train a large model stably in a short time, the same as the conventional DNN.

The CNNN is trained using the CIFAR-10 dataset. Based on the VGG-like CNN structure shown in Fig. 5, which consists of four convolutional layers, two pooling layers, and two fully connected layers, several variations of CNNN with different numbers of neurons are trained. The convolutional layers have eight or 16 channels, which is less than the traditional CNN (e.g., 64 or 128). By optimizing the nonlinear functions, high recognition accuracy can be achieved even when the number of channels is reduced. As described in Section IV-A, average pooling is used in the pooling layers. Since average pooling can be implemented simply by an averaging process, the hardware cost is lower than that of max pooling. By employing the average pooling, there is slight degradation of the recognition accuracy compared with the max pooling. Comparing the accuracy of CNNNs trained with CIFAR-10 using max and average pooling layers, the difference in accuracy is 1%.

Figure 6 shows the training results of the nonlinear functions in each convolution layer, where blue lines indicate linear functions and orange lines indicate non-linear functions. Interestingly, Layers 2 and 3 have many neurons with functions exhibiting a strong degree of non-linearity, while Layers 1 and 4 have many neurons with linear functions.

Figure 7 shows the recognition accuracy dependency on the number of synapses. In this experiment, CNNNs with the same number of neurons are trained with different total numbers of synapses by changing the pruning rate. As the number of synapses increases, the trainable parameters increase, and the expressive ability increases so that the recognition accuracy tends to improve. However, under the condition that the number of neurons and the network structure are fixed, the recognition accuracy saturates at a certain number of synapses. In Sections IV and V, FPGA implementations are performed using three models with different numbers

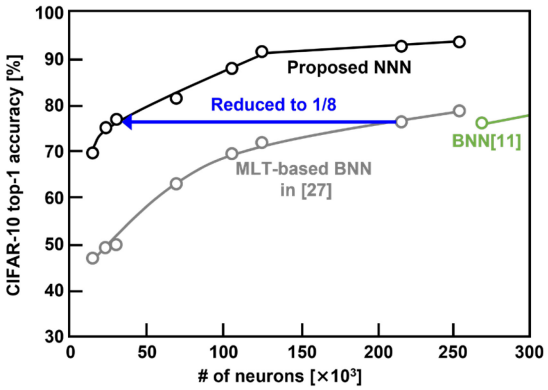


FIGURE 8. Comparison of neurons and accuracy with conventional BNN models.

TABLE 2. Detailed comparison with prior BNN models.

	BNN Ref. [11]	MLT Ref. [27]	Proposed CNNN	BNN Ref. [11]	MLT Ref. [27]	Proposed CNNN
Dataset	CIFAR-10			CIFAR-10		
Accuracy	76.0%	76.7%	75.8%	82.0%	80.0%	81.6%
# of neurons	2.7×10^5 (1)	2.2×10^5 (1/1.2)	2.8×10^4 (1/9.6)	5.4×10^6 (1)	2.6×10^5 (1/21)	2.8×10^4 (1/195)
# of synapses	5.3×10^7 (1)	9.8×10^6 (1/5.4)	1.5×10^5 (1/346)	2.2×10^8 (1)	7.7×10^7 (1/2.9)	4.0×10^5 (1/551)
# of LUTs	2.5×10^8 (1)	4.7×10^7 (1/5.3)	8.5×10^5 (1/295)	1.3×10^9 (1)	4.3×10^8 (1/2.9)	2.4×10^6 (1/528)

of synapses and neurons. The largest model has enough synapses for its recognition accuracy to saturate.

Figure 8 shows the comparison between the simple BNN trained with the MLT method [27] and the CNNN where nonlinear functions of neurons are optimized. In this experiment, the number of neurons is varied while the pruning ratio is fixed at 90%. As shown in Fig. 8, by optimizing the nonlinear functions individually, the number of neurons required to obtain the same recognition accuracy is reduced to 1/8. By optimizing the nonlinear function, the expressive ability of each neuron is greatly improved, and the required number of neurons is significantly reduced.

The proposed CNNN is compared with the state-of-the-art BNN [11] (Table 2), which was implemented as an ASIC and is the most energy-efficient compared with other prior works (as discussed later in Section V). In the conventional BNN [11], the weights are simply binarized using a conventional bit quantization method.

Compared with the conventional NNN [15] (Section III-A), which is trained by reinforcement learning, the proposed CNNN has achieved equivalent neuron and synapse reduction effects. Specifically, in the conventional NNN, the number of neurons is reduced by 1/5.7 and the number of synapses is reduced by 1/274 compared to the conventional BNN for MNIST application, while in the proposed CNNN, the number of neurons is reduced by 1/9.6

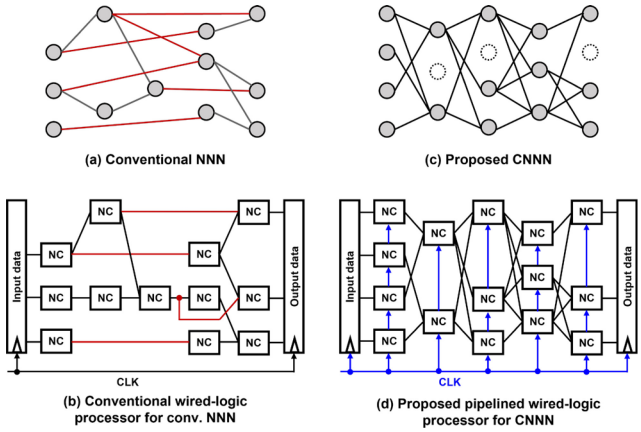


FIGURE 9. (a) (b) Conventional NNN and its wired-logic processor and (c) (d) proposed CNNN and its pipelined wired-logic processor.

and the number of synapses is reduced by 1/346 compared to the conventional BNN for CIFAR-10 application. These results demonstrate that the proposed CNNN method has almost the same optimization effect as the conventional reinforcement learning method. The number of LUTs are estimated for when each NN is implemented using the conventional wired-logic method [15]. As shown in the table, the number of required LUTs is reduced by two orders of magnitude (1/528) because the number of synapses is significantly reduced by NNN technology, enabling implementation on a single FPGA board.

IV. PIPELINED WIRED-LOGIC PROCESSOR

A. PROCESSOR ARCHITECTURE AND IMPLEMENTATION

Unlike DNN, the structure of the conventional NNN is not based on hierarchical layers, and many synapses straddle layers (Fig. 9 (a)). Therefore, in the conventional wired-logic CNNN processor, many wires connect neuron cells (NC) across layers (red lines in Fig. 9). As a result, it is difficult to divide the NNN into a pipelined structure (Fig. 9 (b)), and so the throughput decreases as the scale of the NNN increases.

The proposed CNNN is generated from CNN and inherits its structure, so there are no synapses that straddle layers (Fig. 9 (c)). Therefore, by inserting a flip-flop in each CNN layer (convolution 1st layer, 2nd layer, ...), it can easily be pipelined (Fig. 9 (d)). Each neuron of CNNN is implemented as a neuron cell circuit (NC), and synapses connecting neurons are implemented as wiring between NCs. A neuron cell consists of adder circuits, LUTs, and flip-flops (Fig. 10). Since CNNN uses various nonlinear functions for each neuron, the calculation of nonlinear functions with a simple, small, and energy-efficient circuit is a technical issue.

To address this issue, the activation LUT (Act-LUT) method is utilized, which stores the pre-calculated results of the nonlinear function as a LUT form (the same as the conventional method [15]). This enables the activation functions to be calculated with fewer hardware resources and lower

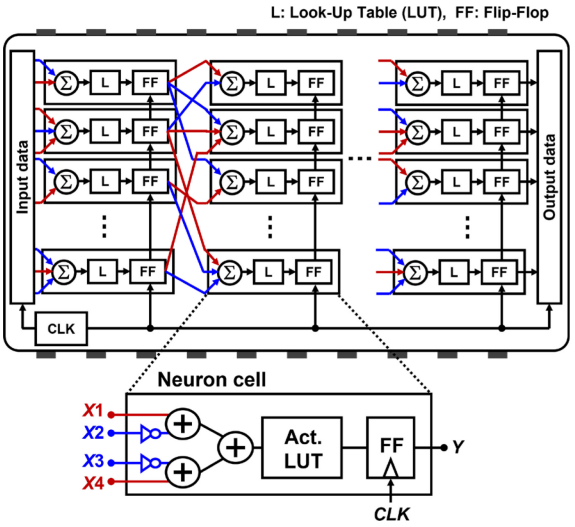


FIGURE 10. Proposed pipelined wired-logic processor.

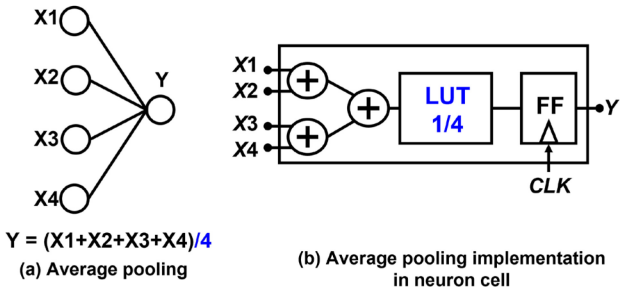


FIGURE 11. Implementation of average pooling layer.

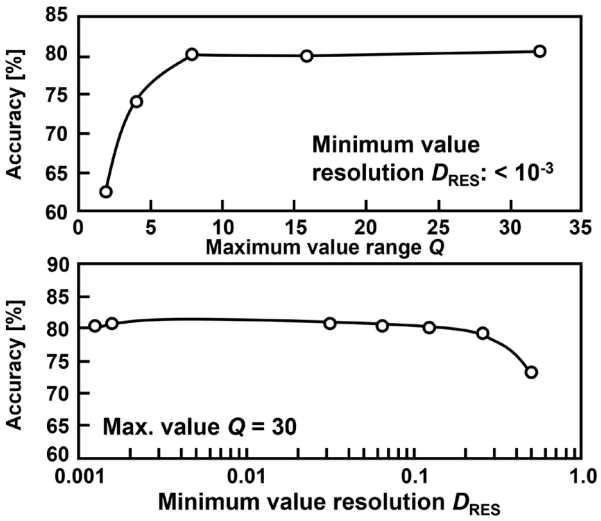


FIGURE 12. Maximum and minimum value ranges vs. accuracy.

power consumption than computing methods using circuits, such as CORDIC. Moreover, the Act-LUTs can be simply implemented with the FPGA LUTs. CNN has additional parameters such as biases and scale factors. To calculate such parameters, each activation function, which includes both a bias factor and a scale factor, is pre-calculated and stored in an Act-LUT.

CNNN inherits the characteristics of BNN, including the binary weights ($+1/-1$). In the wiring with a weight of $+1$, data are directly input to the adder circuit. In the wiring with a weight of -1, data are converted into their complement representations and then input to the adder. The output of Act-LUT is stored in a flip-flop and NC outputs the data in synchronization with the clock signal.

While the conventional NNN [15] has no pooling layers, CNNN inherits the characteristics of CNN, so it has some pooling layers. As discussed in Section III (Fig. 5), average pooling layers are used, each of which consists of adder circuits and LUTs for obtaining the average value, as shown in Fig. 11. For example, the average pooling neuron cell is implemented by receiving four inputs, adding them, and multiplying by 0.25 using an LUT to generate the output. It can be implemented in circuits similar to other neuron cell circuits.

B. BIT WIDTH OPTIMIZATION

Since NNN uses a wide variety of non-linear functions to improve the expressive ability, a sufficient bit width is required. Unlike the conventional BNN, the representation of the intermediate data cannot be binarized.

For all data D generated in CNNN, the value of D is converted into data D' (the data value is clipped at $\pm Q$ (Eq. (2))).

$$D' = \begin{cases} D & (|D| \leq Q) \\ Q & (D > Q) \\ -Q & (D < -Q). \end{cases} \tag{2}$$

Investigation of the recognition accuracy dependency of CNNN on the value of Q (Fig. 12) shows that the recognition accuracy of CNNN degrades to less than 80% when Q is smaller than 8. Next, the sensitivity of recognition accuracy to the numerical resolution of data is investigated. In this experiment, Q is set to 8. Data D' is further converted into data D'' with Eq. (3), and the dependency of the recognition accuracy of CNNN on D_{RES} is investigated

$$D'' = D_{RES} * Round(D'/D_{RES}) \tag{3}$$

As shown in Fig. 12, the recognition accuracy is significantly degraded for $D_{\mathrm{RES}} > 0.25 (= 2^{-2})$.

On the basis of the above results, this work utilizes the INT 6-bit representation, in which the position of the decimal point is placed between the 2nd and 3rd digits from the LSB. The MSB is the sign bit.

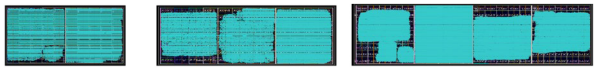
Since the bit width is INT 6-bit, the Act-LUT is composed of six Xilinx FPGA LUTs, and an N-input adder circuit consists of $6 \times N$ Xilinx FPGA LUTs. Most of the FPGA hardware resources are consumed by the adder circuits.

V. EXPERIMENTS AND DISCUSSIONS

The proposed wired-logic processor using CNNN was implemented on an FPGA. As with the conventional method [15],

TABLE 3. Detailed implementation results.

		Small	Middle	Large
Synapses		105,937	131,665	399,090
Neurons		27,713	18,954	27,713
Synapse/neuron ratio		3.8	7.0	14.4
Top-1 accuracy		70.6 %	75.3 %	81.6 %
# of LUTs		540,862	704,463	2,328,307
# of FFs		120,345	112,366	172,988
BRAM		0	0	0
FPGA		VU7P	VU11P	VU19P
Power consumption	Static	1.61 W	2.17 W	8.30 W
	Dynamic	2.15 W	2.78 W	10.00 W
	Total	3.76 W	4.95 W	18.3 W
Frequency		40.0 MHz	40.0 MHz	30.0 MHz
Energy efficiency		0.09 μJ/frame	0.12 μJ/frame	0.61 μJ/frame



(a) Small (b) Middle (c) Large

	Small	Middle	Large
FPGA	VU7P	VU11P	VU19P
LUT usage [%]	68.62	54.36	84.19
FF usage [%]	7.63	4.34	3.01
BRAM usage [%]	0	0	0

FIGURE 13. Implementation results of three types of CNNN.

all Verilog codes were automatically generated from Python. The Xilinx UltraScale+ Virtex series [29] and CIFAR-10 dataset were used.

The implementation results are shown in Fig. 13 and Table 3. Three types of CNNN (Small/Middle/Large) were trained. The network structure was the same as shown in Fig. 5, but the number of synapses and neurons was varied by changing the pruning ratio. Differences between the three types of CNNN are quantified by the factor "synapse/neuron ratio". The factor for small was 3.8, for middle was 7.0, and for large was 14.4. The factor for the large model was twice as large as that for the middle model and four times as large as that for the small model.

The optimum FPGA size was used for each CNNN model. For example, in the small CNNN model, Virtex VU7P was used, and the LUT usage ratio was 68.62 % while the FF usage ratio was 7.63 %. For the middle CNNN model, Virtex VU11P was used, and the LUT usage rate was 54.36 % while the FF usage rate was 4.34 %. For the large CNNN model, Virtex VU19P was used, LUT usage rate was 84.19 % while the FF usage rate was 3.01 %. Note that BRAM is not used (Table 3), as intermediate data are transmitted directly between NCs without being stored in memory.

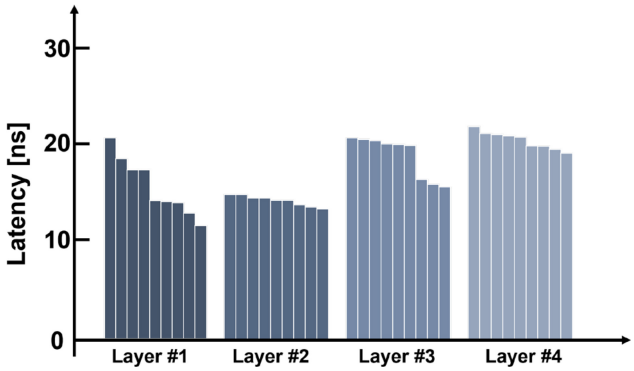


FIGURE 14. Experimental results of latency (large model).

The latency for each layer was measured for each CNNN model and was found to be lower than 23 ns for all three sizes. A part of the measured latency in the large model is shown in Fig. 14. For each layer, nine samples of latency data of different paths are shown. Layers #1 to #4 in Fig. 14 represent the first to fourth layers of the convolution layers as seen from the input, respectively. These measured results of the latency suggest that the operating frequency can be raised up to about 40 MHz with margin. Therefore, for the small and middle CNNN models, the operating frequency was set to the maximum frequency of 40 MHz. For the large one, the operating frequency was reduced so as to ensure a power consumption of 20 W, which is basically the largest power consumption acceptable for edge computing. Consequently, the frequency of the small and middle models was 40 MHz and large one was 30 MHz. The resultant power consumption and energy efficiency for each model are listed in Table III.

The highest energy efficiency was achieved with the small CNNN model (0.09 μJ/frame). As the number of neurons and synapses increases, the recognition accuracy increases, but so too does the number of LUTs. As the number of required LUTs increases, it becomes necessary to implement the CNNN on a larger FPGA. This results in an increased leakage current of the SRAM, an increased static power consumption, and a degraded operating frequency due to the power consumption limit. In other words, energy efficiency degrades as the model size of the CNNN increases. The energy efficiency of the large model with a recognition accuracy of 81.6 % is 0.61 μJ/frame. Owing to the wired-logic architecture, high throughput of 30×10^6 fps is achieved. This makes real-time object recognition possible with ultra-high-speed cameras (e.g., 20 Mfps) [30], [31], which is difficult with prior works due to the long latency of the memory access.

A comparison of the proposed CNNN processor and various state-of-the-art DNN processors optimized for the CIFAR-10 dataset is shown in Table 4. The energy efficiency of the proposed processor is improved by 270 times compared to the FPGA-based state-of-the-art BNN processor [10] and the neuromorphic processor [17], which

TABLE 4. Performance comparison with state-of-the-art works.

	Ref. [11]	Ref. [9]	Ref. [17]	Ref. [12]	Ref. [13]	Ref. [10]	This work
Feature	Digital BNN	Analog BNN	Neuro-morphic	Digital BNN	Digital BNN	Digital BNN	Digital CNNN
CIFAR-10 accuracy	82	86	83	87	89	80	82
Platform	ASIC 28-nm	ASIC 28-nm	ASIC 28-nm	FPGA ZC702	FPGA ZU3EG	FPGA ZC706	FPGA VU19P
Power [W]	1.9×10^{-3}	0.9×10^{-3}	0.2	3.3	4.1	3.6	18.3
Throughput [fps]	486	237	1249	521	2807	21900	30×10^{6}
Energy efficiency [μJ/frame]	3.8	3.8	163.3	6334.0	1460.6	164.4 (1)	0.61 (1/270)

is also a wired-logic processor. Furthermore, the energy efficiency of the proposed processor is about 6.3 times higher than that of the BNN processor implemented on ASIC [9]. The same as for the conventional works in [9], [11], the energy efficiency of the proposed CNNN-based wired-logic processor can be further improved by more than one order of magnitude by implementing it as an ASIC chip [3]. Since the static power consumption of the ASIC chip can be lowered compared with that of the FPGA, the power consumption can be lowered to sub-mW level by reducing the frame rate, which is suitable for always-on AI camera applications [2].

For an input size of about CIFAR-10 ($< 32 \times 32 \times 3$), the NNN can be implemented on the FPGAs available on the market, and provides much better energy efficiency than the conventional AI accelerators. IoT applications on the same network scale as the network for CIFAR-10 are being researched. For example, there is research such on face recognition [11], hand-sign recognition for sign language recognition, and simple object detection [36]. The CNNN developed in this study is also expected to be applicable to similar applications.

On the other hand, it is difficult to apply it to huge neural networks such as those for semantic segmentation and ImageNet. This is because as the input image size increases, the number of LUTs required also increases dramatically. For example, in the case of ImageNet (224×224 pixels), the input image size is much larger than that for CIFAR-10 (32×32 pixels). Even for the simple neural network shown in Fig. 5, the number of neurons and synapses both increase by a factor of 49 ($224^2/32^2$) when the input image size is changed to 224×224. The required number of LUTs increases to 3.4×10^7. Furthermore, the depth direction also needs to be increased, and the number of LUTs required is even higher. With commercial FPGAs, the number of LUTs is clearly insufficient. It is not possible to implement such complicated tasks with the current technology in a single FPGA. New circuit technology is needed to reduce the number of LUTs required.

VI. CONCLUSION

An energy-efficient FPGA-based CNN processor that can process an image from the CIFAR-10 dataset with the energy consumption of just 0.61 μJ per frame is proposed. The proposed processor utilizes a wired-logic architecture in which all the neuron circuits that make up the CNN are implemented on an FPGA chip. Since data are directly transferred between processing elements, they are never written to memory. To reduce the required hardware resources, two core technologies were developed: (1) a CNNN in which the expressive ability of each neuron is improved by optimizing the structure of the neural network and its non-linear activation function individually, and (2) a pipelined neuron cell that utilizes an Act-LUT to process the non-linear function with minimal hardware resources. Three types of neural networks optimized for the CIFAR-10 dataset were implemented on 16-nm FPGAs and processed the image data with a power efficiency of 0.09, 0.12, and 0.61 μJ/frame and a recognition accuracy of 70 %, 75 %, and 82 % respectively. Compared with a state-of-the-art accelerator implemented on FPGA using a binary neural network, the energy efficiency is improved by more than two orders of magnitude.

Since the hardware resource utilization ratio is still large, further technology improvements to reduce the network size and circuit area will be required. In particular, since the large model prototyped in this work utilized the largest FPGA currently on the market (Virtex VU19P), a more efficient implementation is desired in order to lower the costs.

REFERENCES

[1] A. Kosuge, K. Yamamoto, Y. Akamine, and T. Oshima, "An SoC-FPGA-based iterative-closest-point accelerator enabling faster picking robots," *IEEE Trans. Ind. Electron.*, vol. 68, no. 4, pp. 3567–3576, Apr. 2021.

[2] P. Jokic, E. Azarkhish, R. Cattenoz, E. Türetken, L. Benini, and S. Emery, "A sub-mW dual-engine ML inference system-on-chip for complete end-to-end face-analysis at the edge," in *IEEE Symp. VLSI Circuits Dig. Tech. Papers*, Kyoto, Japan, Jun. 2021, pp. 1–2.

[3] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuit Conf. (ISSCC), Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2014, pp. 10–14.

[4] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.

[5] W. J. Gross, B. H. Meyer, and A. Ardakani, "Hardware-aware design for edge intelligence," *IEEE Open J. Circuits Syst.*, vol. 2, pp. 113–127, 2021.

[6] Y. Chen, L. Lu, B. Kim, and T. T.-H. Kim, "A reconfigurable 4T2R ReRAM computing in-memory macro for efficient edge applications," *IEEE Open J. Circuits Syst.*, vol. 2, pp. 210–222, 2021.

[7] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR) Dig. Tech. Papers*, Long Beach, CA, USA, Jun. 2019, pp. 8604–8612.

[8] K. Ando *et al.*, "BRein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, Apr. 2018.

[9] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC), Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 222–224.

[10] Y. Umuroglu *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, Feb. 2017, pp. 65–74.

[11] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, "BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28nm CMOS," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, San Diego, CA, USA, Apr. 2018, pp. 1–4.

[12] M. Ghasemzadeh, M. Samragh, and F. Koushanfar, "ReBNet: Residual binarized neural network," in *Proc. IEEE 26th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Boulder, CO, USA, Apr. 2018, pp. 57–64.

[13] Y. Zhang, J. Pan, X. Liu, H. Chen, D. Chen, and Z. Zhang, "FracBNN: Accurate and FPGA-efficient binary neural networks with fractional activations," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2021, pp. 171–182.

[14] J. T. Welch and J. Carletta, "A direct mapping FPGA architecture for industrial process control applications," in *Proc. Int. Conf. Comput. Design*, Austin, TX, USA, Sep. 2000, pp. 595–598.

[15] A. Kosuge, M. Hamada, and T. Kuroda, "A 16 nJ/classification FPGA-based wired-logic DNN accelerator using fixed-weight non-linear neural net," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 4, pp. 751–761, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9541384

[16] A. Krizhevsky, "Convolutional deep belief networks on CIFAR-10," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Rep. TR 2010, Aug. 2010. [Online]. Available: https://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf

[17] S. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci.*, vol. 113, no. 41, pp. 11441–11446, Oct. 2016.

[18] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[19] M. Koo, G. Srinivasan, Y. Shim, and K. Roy, "sBSNN: Stochastic-bits enabled binary spiking neural network with on-chip learning for energy efficient neuromorphic computing at the edge," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 8, pp. 2546–2555, Aug. 2020.

[20] J. M. Correll *et al.*, "A fully integrated reprogrammable CMOS-RRAM compute-in-memory coprocessor for neuromorphic applications," *IEEE J. Explor. Solid-State Computat.*, vol. 6, no. 1, pp. 36–44, Jun. 2020.

[21] A. Neckar *et al.*, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proc. IEEE*, vol. 107, no. 1, pp. 144–164, Jan. 2019.

[22] Y. Sakemi, K. Morino, T. Morie, and K. Aihara, "A supervised learning algorithm for multilayer spiking neural networks based on temporal coding toward energy-efficient VLSI processor design," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 19, 2021, doi: 10.1109/TNNLS.2021.3095068.

[23] A. Gaier and D. Ha, "Weight agnostic neural networks," 2019, *arXiv:1906.04358*.

[24] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002.

[25] A. Mirhoseini *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, pp. 207–212, Jun. 2021.

[26] J. Frankle and M. Carbin "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2019, *arXiv:1803.03635*.

[27] J. Diffenderfer and B. Kailkhura, "Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network," 2021, *arXiv:2103.09377*.

[28] N. Ma *et al.*, "Activate or not: Learning customized activation," 2021, *arXiv:2009.04759*.

[29] "VIRTEX® UltraSCALE+." Xilinx Inc. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/virtex-ultrascale-plus.html (Accessed: Sep. 21, 2021).

[30] M. Suzuki *et al.*, "An over 1Mfps global shutter CMOS image sensor with 480 frame storage using vertical analog memory integration," in *IEEE Int. Electron Devices Meeting (IEDM) Dig. Tech. Papers*, San Francisco, CA, USA, Dec. 2016, pp. 1–4.

[31] Y. Tochigi *et al.*, "A global-shutter CMOS image sensor with readout speed of 1-Tpixel/s burst and 780-Mpixel/s continuous," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 329–338, Jan. 2013.

[32] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. ICLR*, Apr. 2017, pp. 1–18.

[33] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. ICLR*, Apr. 2017, pp. 1–16.

[34] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 19–35.

[35] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 1761–1770.

[36] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

ATSUTAKE KOSUGE (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Keio University, Yokohama, Japan, in 2012, 2014, and 2016, respectively.

From 2014 to 2017, he was a JSPS Research Fellow with Keio University. From 2017 to 2020, he held research positions with Hitachi Ltd., and Sony Corporation. In 2021, he joined the University of Tokyo, where he is currently an Assistant Professor of Systems Design Lab. His research interests include energy efficient computing, computational sensing, and 3-D integration technologies.

Dr. Kosuge was the recipient of the 2013 Nikkei Electronics Japan Wireless Technology Best Award, the 2020 IEICE Young Researcher's Award, and co-recipient of the ASP-DAC'15 Special Feature Award. He has been serving as a member for the Technical Program Committee of IEEE Asian Solid-State Circuits Conference since 2021, IEICE Integrated Circuit and Devices, and served as a member for the Organizing Committee of IEEE COOL Chips (Symposium on Low-Power and High-Speed Chips and Systems).

YAO-CHUNG HSU received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2019. He is currently pursuing the M.S. degree with the University of Tokyo, Tokyo, Japan.

His current research interest includes energy-efficient integrated circuit design for emerging machine learning applications.

MOTOTSUGU HAMADA (Member, IEEE) was born in Nara, Japan, in 1968. He received the B.S., M.S., and Ph.D. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1991, 1993, and 1996, respectively.

In 1996, he joined Toshiba Corporation and was engaged in wireless and low-power electronic circuits design with Toshiba's Center for Semiconductor Research and Development, Kawasaki, Japan. From 2002 to 2004, he was a Visiting Scholar with Stanford University. From 2011 to 2016, he was with Mixed Signal IC Division as a Group Manager of Power Analog IC Design Group to lead the development of analog mixed signal ICs. In 2016, he joined Keio University and was a Project Professor. In 2020, he joined the University of Tokyo, where he is currently a Project Professor of Systems Design Lab. His research interests include low-power, high-speed CMOS design, low-power wireless systems and circuits design, and power management systems design.

Dr. Hamada was the recipient of the 2007 IEEE International Conference on Computer Design (ICCD) Best Paper Award and the Design Automation Conference (DAC) 2010 Best User Track Poster Award. He was also recognized in the list of "Authors of Ten or More Papers in the Past Ten Years" at International Solid-State Circuits Conference in 2013 (ISSCC2013). He has served as a member of the technical program committee of International Solid-State Circuits Conference from 2003 to 2009 and in 2011 and VLSI Circuits Symposium from 2018 to 2021, and Asian Solid-State Circuits Conference from 2005 to 2012 and from 2017 to 2021, where he served as a RF Subcommittee Chair, a Digital Subcommittee Chair, a Student Design Contest Chair, and a Technical Program Committee Chair.

TADAHIRO KURODA (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1999.

In 1982, he joined Toshiba Corporation. From 1988 to 1990, he was a Visiting Scholar with the University of California at Berkeley, where he conducted research in the field of VLSI CAD. In 1990, he was back to Toshiba, and engaged in the research and development of BiCMOS ASICs, ECL ASICs, and high-speed low-power CMOS LSIs. He invented a variable threshold-voltage CMOS (VTCMOS) technology to control V_{TH} through substrate bias, and applied it to a DCT core processor in 1995. He also developed a variable supply-voltage scheme to control V_{DD} by an embedded DC-DC converter, and employed it to a microprocessor core and an MPEG-4 chip in 1997. He left Toshiba to join Keio University in 2000, and became a Full Professor in 2002. He was the Mackay Professor with the University of California at Berkeley, in 2007. He invented a ThruChip Interface by using magnetic coupling for communications among stacked chips in 2008, and a Transmission Line Coupler by using electromagnetic coupling for communications among stacked PCBs in 2012. He left Keio to join the University of Tokyo in 2019. He is the Director of Systems Design Lab with the University of Tokyo, and the Chairman of Research Association for Advanced Systems. He has published more than 450 papers, including 38 ISSCC papers, 29 VLSI Symposia papers, 19 CICC papers, and 18 A-SSCC papers. He wrote 30 books/chapters and filed more than 200 patents.

Prof. Kuroda was a recipient of the 2005 P&I Patent of the Year Award, the 2007 ASP-DAC Best Design Award, the 2009 IEICE Achievement Award, and the 2011 IEICE Society Award. He served as a Steering Committee Chair for A-SSCC, the Vice Chair for ASP-DAC, the Sub-Committee Chairs for A-SSCC, ICCAD, SSDM, and VLSI-DAT, and the TPC Member for ISSCC, Symposium on VLSI Circuits, CICC, DAC, ASPDAC, ISLPED, SSDM, ISQED, and other international conferences. He is the chair of Symposium on VLSI Technology and Circuits. He is an IEICE Fellow. He was an elected AdCom Member of two terms. He was a Distinguished Lecturer and a representative of Region 10 for the IEEE Solid-State Circuits Society.