PAPER Special Section on Solid-State Circuit Design — Architecture, Circuit, Device and Design Methodology

SLIT: An Energy-Efficient Reconfigurable Hardware Architecture for Deep Convolutional Neural Networks

Thi Diem TRAN[†a)], ***Nonmember and*** **Yasuhiko NAKASHIMA**[†], ***Fellow***

SUMMARY Convolutional neural networks (CNNs) have dominated a range of applications, from advanced manufacturing to autonomous cars. For energy cost-efficiency, developing low-power hardware for CNNs is a research trend. Due to the large input size, the first few convolutional layers generally consume most latency and hardware resources on hardware design. To address these challenges, this paper proposes an innovative architecture named SLIT to extract feature maps and reconstruct the first few layers on CNNs. In this reconstruction approach, total multiply-accumulate operations are eliminated on the first layers. We evaluate new topology with MNIST, CIFAR, SVHN, and ImageNet datasets on image classification application. Latency and hardware resources of the inference step are evaluated on the chip ZC7Z020-1CLG484C FPGA with Lenet-5 and VGG schemes. On the Lenet-5 scheme, our architecture reduces 39% of latency and 70% of hardware resources with a 0.456 W power consumption compared to previous works. Even though the VGG models perform with a 10% reduction in hardware resources and latency, we hope our overall results will potentially give a new impetus for future studies to reach a higher optimization on hardware design. Notably, the SLIT architecture efficiently merges with most popular CNNs at a slightly sacrificing accuracy of a factor of 0.27% on MNIST, ranging from 0.5% to 1.5% on CIFAR, approximately 2.2% on ImageNet, and remaining the same on SVHN databases.

key words: *primary visual cortex, image classification, convolutional neural network, hardware architecture, FPGA, feature extraction*

1. Introduction

Convolutional neural networks (CNNs) have many prominent applications with superior results such as object detection, image classification, robot vision [1]–[3]. However, their complicated network architecture and power consumption, which have adverse influences on latency and required accuracy, have posed many challenges. The CNNs use the forward stage for inference and the feed-backward stage for training. Training of deep networks often requires significant resource, energy, and computation time. Many authors have chosen the off-line training for CNNs in practical applications or used the trained model to perform accelerators [4], [5]. How to speed up the feed-backward performance and the feed-forward stage is a critical concern on CNN. In other words, it is desirable to seek an efficient optimization methodology, which can guarantee accuracy with the least loss.

Manuscript received May 28, 2020.
Manuscript revised November 7, 2020.
Manuscript publicized December 18, 2020.

[†]The authors are with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
a) E-mail: tran.thi_diem.tl4@is.naist.jp

DOI: 10.1587/transele.2020CDP0002

Researchers have investigated to optimize memory access or convolution operations. Recent works showed that the sparsity optimization that involved pruning and exploiting activation sparsity could reduce 89% of memory access and 67% of computation operations [6]. Activation sparsity can cut memory accesses and multiply-accumulate (MAC) operations by a half [7] based on rectified linear unit non-linearity to produce many zero outputs. The pruning and compression were also investigated with the Bayesian network. These reductions have nevertheless required the hardware that was customized with the data movement and control. The reducing parameter approaches [8], [9] with a factor of 50× were studied due to the expense of many MAC operations. The low-rank approximation (LRA) that obtained a sparse convolution 2× - 4.5× faster than the corresponding value at the absence of sparsity with a 1% accuracy loss was reported by Denton et al. [10]. Due to a large number of hyper-parameters, the LRA has remained to be a big problem in training. The bit-width optimization that aims to decrease the bit-width of parameters from a floating-point to a fixed point is another approach. This investigation reduced the precision to get higher efficiency in exchange for memory access and computation operations. Ternary weight network and BinaryConnect [11], [12] are examples of the bit-width reduction of weights to 2-bits or 1-bit. Some studies also quantized the activation function of the neural network, which achieved a significant reduction in memory or computation cost [13], [14]. These proposals nevertheless were a trade-off for a considerable accuracy loss. As a result, the gain of compact network architecture is the loss of accuracy.

The main challenges in using CNNs are latency and memory access [15], [16], due to tens to hundreds of megabyte parameters and operations which require data movement between on-chip and off-chip to support the computation. In edge applications such as smart sensors, wearable and autonomous devices, security and latency are important considerations [17], [18]. We have recently surveyed the performance of state-of-the-art CNNs in terms of accuracy, size, and potentiality of various hardware platforms. The results reveal a gap between the designers who strike for comprehensive CNNs with better efficiency and the hardware architects who try to simplify them [19], [20]. Many researchers have attempted to speed up the CNN performance by using graphical processing units (GPU) [21], [22]; yet, the power consumption on GPU remains a critical issue. Moreover, the computation is subject to rigorous area and power constraints in the inference stage due to the lim-

Copyright © 2021 The Institute of Electronics, Information and Communication Engineers

ited available resources. Therefore, many data scientists are focusing on increasing inference performance by designing various accelerators.

Field Programmable Gate Arrays (FPGAs) have become the best candidate for the trade-off of cost, flexibility, and performance in deep learning processor designs [23]. FPGAs are suitable for computationally intensive algorithms that result in a faster speed and efficient energy. A few highlights of these approaches include parameter reduction, binary weight quantization, memory bandwidth optimization, and data-flow optimization [24]–[27]. A highly flexible architecture that can mold itself into the given CNNs and achieve a higher resource utilization reduction is essential. Moreover, due to the large input size, the first few layers that typically contribute to the most significant latency on CNN leave a plenty of room for improvement.

This paper proposes an innovative algorithm for feature extraction and reconfigures some first few layers on CNNs. According to the results, this method proves latency, hardware resources, power consumption, and training time reduction on the conventional CNNs for image classification application. Accuracy and performance are evaluated by using the MNIST [28], SVHN [29], CIFAR-10, CIFAR-100 [30], and ImageNet [31] databases with Lenet-5 and VGG models. The proposal achieves a 39% reduction in latency and a factor of 50% hardware resource deduction on the IP core for the Lenet-5 model compared to the works [32], [33] by using the Vivado_HLS tool. Our accelerator also decreases 70% area at a 0.456 W power consumption, which is less than Refs. [34]–[36] on the chip ZC7Z020 FPGA. Furthermore, the training time is decreased by 40%, 40% and 32%, corresponding with MNIST, CIFAR, and SVHN datasets on Lenet-5 and CNN models. It also reduces by approximately 10% on VGG architectures with the CIFAR database. In summary, our research makes the following contributions.

- A new efficient topology to extract features of input data in the deep neural network is proposed. We have successfully demonstrated how to replace this method for the first few layers on CNNs.
- A new re-configurable CNN for the Lenet-5 and VGG models in image classification application is proposed and evaluated on MNIST, CIFAR-10, CIFAR-100, ImageNet and SVHN datasets.
- We have succeeded in removing convolution operations in the first few layers, which take much latency on CNNs as a bottleneck when implementing CNNs on the hardware platform.
- A hardware architecture with high speed and efficiency energy for the inference phase of the deep neural network is demonstrated.

The rest of this paper is arranged as follows: Sect. 2 reviews the preliminary convolutional neural network. In Sect. 3, the proposed architectures are presented. The methodologies of the verification are manifested in Sect. 4. In Sect. 5, the results of the software and hardware proposal are investigated. Finally, the report is wrapped up with our future research plan in Sect. 6.

2. Preliminary Convolutional Neural Network

Through development over 20 years, the network that was initially inspired by neuroscience has attracted spacious attention in such fields as image processing and computer science [37]–[39]. Today, some object recognition systems based on CNN can recognize objects with super-human accuracy. CNN perceives an object through the feature extraction step and the classification phase. In Fig. 1, the feature extraction step including the convolutional and sub-sampling layers is a part to find variances of an input image such as lines and edges. The classification phase combining the fully-connected (FC) layers decides the most likely class object based on the extracted features. Using the convolutional (CONV), sub-sampling, and FC layers, CNN can achieve a highly accurate classification.

The CONV layer receives features as input and performs convolution operation with a filter kernel window to generate one pixel in one output feature map. The output channels are filtered through an activation function such as Relu, Sigmoid, and Tanh. Total output feature maps form a set of the input channels for the next CONV layer. Summary of the process which calculates one output channel is formulated in Eq. (1).

$$O_j^k = f(\Sigma_{i \in M} I_i^{k-1} * W_{ij}^k + b_j^k) \tag{1}$$

where O_j^k is the current output of the j^{th} channel at k^{th} layer, I_i^{k-1} is the previous feature map of the i^{th} channel in M input channels, W is the ij^{th} kernel filter, b_j^k is corresponding the bias of the j^{th} channel, f is the activation function, and the symbol "$*$" is the element-wise multiplication operation.

The sub-sampling layer or the pooling layer is generally sandwiched between two CONV layers. The pooling layer reduces the size of feature maps from the previous layer. Besides, this layer is employed to avoid the overfitting problem and redundancy in the channels. There are two main pooling methods: mean-pooling and max-pooling. The output of the max-pooling (MP) layer is determined by using Eq. (2).

$$u_{i,j}^m = \max_{0<=i,j \in P} u_{(i,P+i),(j,P+j)}^n \tag{2}$$

where u^m is the max output value in the kernel size P of the m^{th} channel, the u^n is input value in the kernel size P.

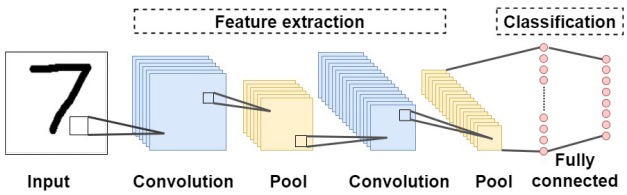

Fig. 1 The general CNN architecture

Algorithm 1 Laplacian filter

```
for i = 1 toN do
    for j = 1 toN do
        c[0] ← in[i − 1][j]
        c[1] ← in[i][j − 1]
        c[2] ← in[i][j]
        c[3] ← in[i][j + 1]
        c[4] ← in[i + 1][j]
        d ← (c[2] ∗ 4 − (c[0] + c[1] + c[3] + c[4]))
        out[i][j] = d < threshold?0 : 255
    end for
end for
```

The FC layers that control object classification into various categories in CNNs are conjoined after multiple convolutional and sub-sampling layers. The term "fully connected" means that all neurons in the previous layer are connected to all neurons in the next layer. For example, the last layer of the Lenet-5 for classifying the MNIST database has ten possible outputs, and each output corresponds to a number from "0" to "9". A neuron output V_k^{out} in the FC layer is obtained by using Eq. (3). It is a typical matrix multiplication and addition with a bias.

$$V_k^{out} = \Sigma_{i=0}^{N} W_{ki} \times V_i^{in} + bias_k \tag{3}$$

where W_{ki} is weights corresponding with N input neurons at k^{th} position, V_i^{in} is the total neurons of the previous layer, and $bias_k$ is the bias of k^{th} output neuron.

3. Proposed Design Optimization

Creating a paradigm that reduces most hardware resources and computation time in the first layers on CNNs without cutting accuracy is the aim of our proposal. We propose an innovative feature extraction layer of data inputs named the SLIT layer to approach this aim. Besides replacing the first CONV layer on CNNs, this layer also conveys a creative alternative to optimize some next layers. Guaranteeing accuracy at allowing threshold, increasing speed, and enhancing convergence are the robust features of this layer.

3.1 Motivation

The proof-of-concept of replacing the first few layers on CNNs is inspired via the primary visual cortex principle [40], [41]. Edge detection plays a vital role in feature extraction. To reach the goal of replacing the first layer on CNNs, we use a filter kernel based on the Laplacian filter to execute edge detection. Algorithm 1 illustrates what we are going to use in our proposal. A 3×3 window is sliced on the N×N image to obtain the result d. Then the edge output is taken by comparing d with a threshold. There is an edge if d value is larger than the threshold; otherwise, it is no edge. After experimenting, we choose 30 as the value threshold to gain the best accuracy.

We propose the shift circuit in a range of 0° to 157.5° with a gradual increase every 22.5° in the 4×4 window, as shown in Fig. 2 [42]. The results of SLIT detection execute at the base of edge detection. The output channel ch_k resulting from the AND function of input edge values at examining slope is computed by applying Eq. (4). Each element is 1-bit, and the output values are 8-bits equivalent to the input values. Concatenating ch0 to ch7, we get eight extracted feature maps to form the SLIT layer.

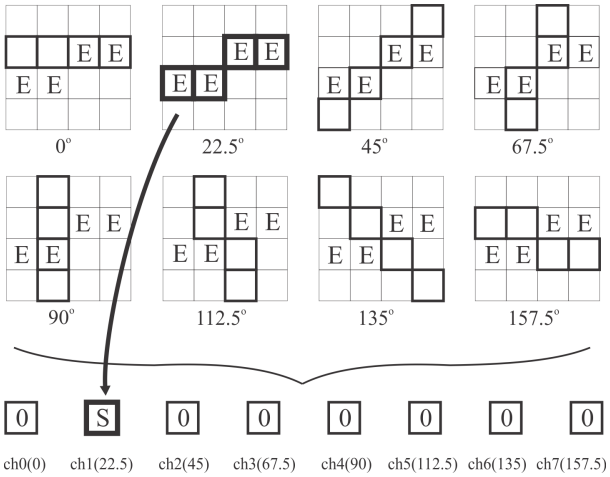

Fig. 2 The SLIT detection with a 4×4 window

Algorithm 2 SLIT layer

```
for i = 1 toN − 1 do
    for j = 1 toN − 1 do
        for k = −1 to 3 do
            for l = −1 to 3 do
                edge[k + 1][l + 1] ← in[i + k][j + l]
            end for
        end for
        db[0] ← edge[1][0]&edge[1][1]&edge[1][2]&
        edge[1][3]&1
        .....
        db[7] ← edge[1][0]&edge[1][1]&edge[2][2]&
        edge[2][3]&1
        for k = 0 to 7 do
            out[k][i][j] ← db[k]
        end for
    end for
end for
```

$$ch_k = AND(\Sigma_{i=0}^{3}\Sigma_{j=0}^{3}\theta_{i,j}) \tag{4}$$

where θ is a steady increase every 22.5°, $\theta_{i,j}$ is the values at examining slope θ in the 4×4 window.

3.2 SLIT Layer Architecture

In many CNNs such as VGG, MobileNet, ResNet, or GoogleNet, the CONV layer is always the first layer. This layer typically performs a whole number of sliding convolution operations due to the largest input size. The first layer requires much computation time when being compared with other layers. In the CONV layer, with a number of input channels (ICs) and the K×K filters, there compute six consecutive loops for producing output channels

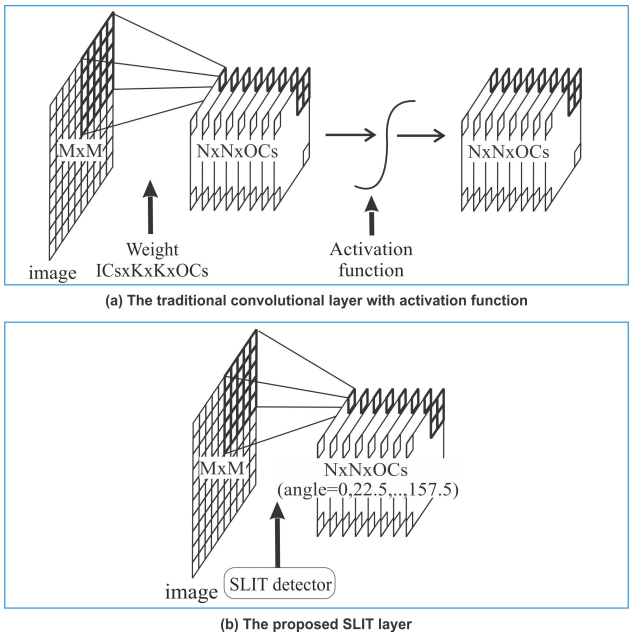

Fig. 3 The proposed SLIT layer

(OCs) in the traditional approach. In contrast, the proposed SLIT layer presented in Algorithm 2 contains only four continuous loops. Figure 3 (a) explains how to calculate the first CONV layer with the traditional approach. The first CONV layer with an M×M input image is convoluted with ICs×K×K×OCs kernel filters to yield N×N×OCs output channels. Subsequently, the Relu activation function is employed to normalize the output values into a range between 0 and 1. In Fig. 3 (b), to obtain N×N×OCs output feature maps like the first CONV layer, we leverage the SLIT layer, as explained beforehand in the motivation section. Due to the binary output, the activation function is discarded after the SLIT layer.

In comparison with the first CONV layer on the original CNNs, the MAC operation and activation function are eliminated in the proposal. Each input is reused across all filters of different output channels within the same layer in the CONV layer. Therefore, storing memory and power consumption have become enormous. On the other hand, since there are no parameters required for the SLIT layer during the training phase and inference step, memory access and latency are significantly reduced in the proposal. The normalization step for inputs, which are divided by 255, is also ignored in our idea. We only use the Shift, AND, and comparator operations to extract feature maps. Consequently, our approach decreases many resources, latency, and energy. Total parameters (*params*) are presented in Eq. (5), and MAC operations (*MACs*) are shown in Eq. (6) are ricocheted in the way that reconstructs with the SLIT layer.

$$params = C\_in \times K \times K \times C\_out \tag{5}$$

where C_in is the number of input channel, K is the size of kernel filter and C_out is the number of output channel.

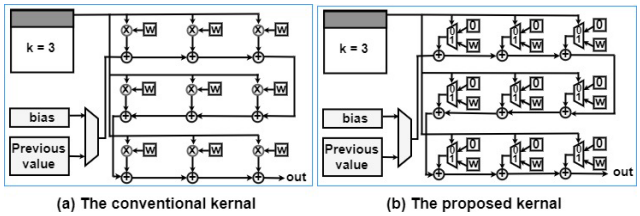

Fig. 4 The proposed kernel for the second layer

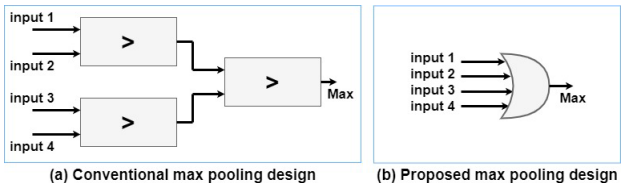

Fig. 5 The proposed max pooling kernel

$$MACs = C\_in \times K \times K \times N\_in \times N\_in \times C\_out \tag{6}$$

where C_in is the number of the input channel, K is the size of kernel filter, N_in is the dimension of output channel, and C_out is the number of output channel.

3.3 Next Layer Reconfiguration

Due to the binary output of the SLIT layer, we propose a new scheme to reconfigure the second CONV, max-pooling (MP), and fully connected (FC) layers. We name SCONV, SMP, and SFC layers for the proposed second CONV, MP, and FC layers. MAC operations also occupy most computation time in the second CONV layer, directly following the first CONV layer on CNN. Many works have been investigated to optimize MAC operations, such as using XOR functions [13]. In contrast, we suggest the architecture that employs multiplexer (MUX) operation to determine output feature maps for the second CONV layer. In Fig. 4 (a), we illustrate the process with a 3×3 kernel filter. To receive the second CONV output channel, there require 9 multiplication operations. On the other hand, our proposal only uses 9 MUX operations to generate a feature map for the second CONV layer showed in Fig. 4 (b).

Our proposal excretes all or a part of multiplication operations in the second CONV layer. First, the complete replacement will affect the situation if the SLIT layer only generates eight binary output feature maps in which the input image has one channel like the MINST database. Second, a part of the replacement will take place when the input image has three channels, such as CIFAR, SVHN, or ImageNet database. The SLIT layer yields 11 channels by concatenating eight binary output feature maps of SLIT function with three original channels normalized in range 0 and 1. In this case, output channels of the second CONV layer are determined by concatenating eight binary output channels of the SLIT layer with three feature maps of the normalized input image.

The max-pooling (MP) layer mentioned in Fig. 5 is

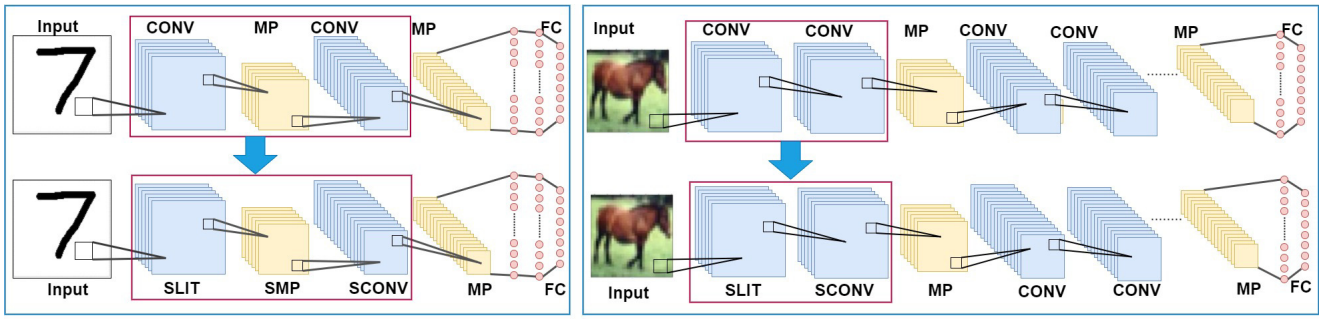

Fig. 7 The proposed Lenet-5 and VGG model

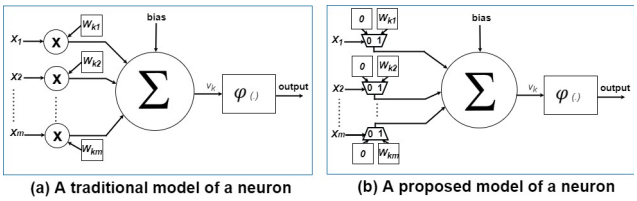

Fig. 6 The proposed model of a neuron

optimized by utilizing an OR gate to determine maximum value. Figure 5 (a) reveals that at least three comparator operations are required to detect the maximum value in the 2×2 window at a stride of 2 with the conventional approach. In contrast, our proposed circuit showed in Fig. 5 (b) only uses the OR gate to estimate the maximum value for the MP layer. Assuming that there have eight 14×14 output channels from the previous layer, a total of 14×14×3×8 = 4704 comparator operations are expected by applying Eq. (7). On the other hand, our proposal demands 14×14×8 = 1568 OR gates with four inputs.

$$Comp = N \times N \times (K \times K - 1) \times Cout \tag{7}$$

where $Comp$ is total comparator operations required to determine the maximum value, N is the size of the previous channel, K is the kernel size, and $Cout$ is the number of output channels.

The FC proposed layer is affected by a model that consolidates the previous CONV layer and an MP layer. The basic information processing unit of one neural in the artificial network is demonstrated in Fig. 6 (a). The inputs are multiplied with corresponding weights, and then outcomes are added with a bias. Figure 6 (b) shows the matrix multiplication replaced by the MUX operations, where the weight values are one. Equation (8) defines the entire multiplication operations, which occupy most time consumption in the FC, are reduced by the multiplexer operations. Assume that we have 1024 input neurons and 1024 output neurons; by using Eq. (8), the entire 1024×1024 = 1M multiplication operations are pruned.

$$Muls = Num\_in \times Num\_out \tag{8}$$

where $Muls$ is total multiplication operation to calculate one output. Num_in is the entire input neurons and Num_out is the whole output neurons.

3.4 Complete Proposed System

This section demonstrates how to reconstruct the first two layers and the first three layers of our proposal in practical applications. We choose the Lenet-5 that combines one CONV + MP + another CONV layers in the model to manifest how to reconfigure with SLIT + SMP + SCONV layers. In Fig. 7 (a), we replace the CONV + MP + CONV layers with SLIT + SMP + SCONV layers. The remaining layers stay the same as the original model. Figure 7 (b) shows how to reconfigure the first two CONV layers that occur in some famous models such as VGG-16, VGG-19. These models include two CONV layers before the MP layer. In this fashion, the first two CONV layers are changed by SLIT and SCONV layers.

4. Experimental Setup

In this section, we investigate the utilization of the SLIT layer in various models. We conduct extensive experiments on the standard Lenet-5, VGG-16, and VGG-19 prototypes with the MNIST, SVHN, CIFAR-10, CIFAR-100, and ImageNet datasets. We use Tensorflow, Keras, and Pytorch platforms to build the models. Training time and accuracy of the MNIST, CIFAR, and SVHN databases are analyzed by using the Intel(R) Core(TM) i7-3970X CPU @ 3.50GHz. We use the GeForce GTX 1080 for training the ImageNet dataset. To determine hyper-parameter values in our model, we first train the examining database on the traditional model to estimate the hyper-parameters at an acceptable accuracy like benchmarks. Then, during the training phase of the proposed model, we increase or decrease appropriately the value of the reference hyper-parameters extracted from the conventional model. Finally, hyper-parameters of the proposal are determined when the over-fitting and under-fitting phenomena disappear, and the model converges with the highest accuracy. In comparison with the traditional model at the same database, the hyper-parameters are nearly identical between the two models. Therefore, we have used the same values when training

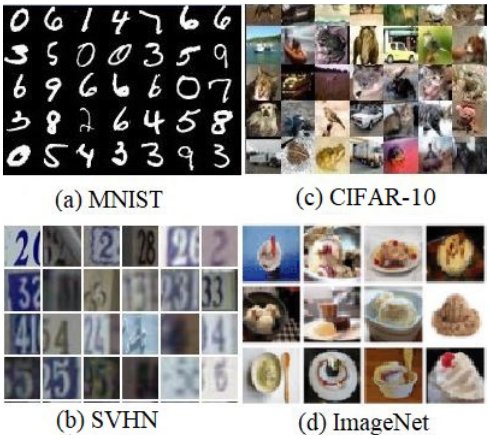

Fig. 8 The examples of MNIST, CIFAR, SVHN and ImageNet databases

conventional and proposed models. We evaluate latency, hardware resources, and power consumption of the inference phase on the chip ZC7Z020-1CLG484C FPGA.

4.1 Software Configuration

Handwritten digits (MNIST): The MNIST database [28] consists of 28×28 gray images of the handwritten digits "0" through "9". A total of 60000 images are provided for training, and 10000 images leave for testing. In the reported experiment, training images are sliced further into a training set (50000 images) and a validation set (10000 images), equal to the distribution of digit classes. Figure 8 (a) shows samples of the MNIST dataset. The Lenet-5 paradigm is considered for performance analyses. From the original Lenet-5 model which combines CONV(6) + MP(2) + CONV(16) + MP(2) + FC(120) + FC(84) + FC(10), we propose the design that mixes of SLIT(8) + SMP(2) + SCONV(16) + MP(2) + FC(120) + FC(84) + FC(10). The simulation is carried out using a batch size of 100 images, 20 epochs, and the stochastic gradient descent (SGD) optimization function with a learning rate of 0.1.

SVHN dataset: We examine our models on the SVHN dataset [29], which has three channels in an image. SVHN is collected from house numbers in Google Street View images. It includes 73257 images for training and 26032 images for testing. Examples of the SVHN dataset are displayed in Fig. 8 (b). We investigate the SVHN dataset with the model that combines 2CONV(32) + MP(2) + 2CONV(64) + MP(2) + FC(512) + FC(10). In this manner, we replace two CONV(32) layers with SLIT(11) and SCONV(32) layers. The model is trained with a batch size of 128 images, 20 epochs, and the SGD optimization function with a learning rate of 0.01.

CIFAR database: The proposal is interpreted in detail on the CIFAR-10 and CIFAR-100 datasets [30]. These datasets are composed of 60000 samples from ten categories for CIFAR-10 and 100 categories for CIFAR-100. Figure 8 (c) shows examples of the CIFAR-10 dataset. We utilize 45000 images for training, 5000 images for validation and the last 10000 images for testing, and augment the database by exerting flip and shift operators. The Lenet-5, VGG-16, and VGG-19 models are employed for measuring performance. These models are assessed with a batch size of 128 samples, 200 epochs, and the SGD optimization function with learning rate change from 0.1 in a range of 0 to 100 epochs, 0.01 in a range of 100 to 150 epochs, and 0.001 for larger than 150 epochs. Since the CIFAR dataset has three input channels, we concatenate eight output feature maps of the SLIT function with three input channels normalized in a range 0 and 1 to create the SLIT layer. For the Lenet-5 design, we validate with SLIT(11) + SMP(2) + SCONV(16) + MP(2) + FC(120) + FC(84) + FC(10) as the equivalence of the conventional Lenet-5 scheme which stacks up of CONV(6) + MP(2) + CONV(16) + MP(2) + FC(120) + FC(84) + FC(10). In the VGG-16 and VGG-19 forms, two first CONV layers with 64 output channels are switched by SLIT(11) + SCONV(64) layers.

ImageNet database: We have chosen the ILSVRC2012 ImageNet dataset [31] as a target to assess our topology in a complicated case. ImageNet includes approximately 1.2M training images with 1K classes and 50K validation images. This dataset covers natural images with reasonably high resolution compared to the CIFAR, MNIST, and SVHN datasets, which have relatively small images. The examples of the ImageNet database are shown in Fig. 8 (d). We conduct our image classification performance to report Top-1 and Top-5 accuracy. We adopt VGG-16 architecture as our base proposal. Two first CONV layers of the VGG-16 model is reconstructed with SLIT(11) and SCONV layers. The design is simulated with a batch size of 16 samples, 100 epochs, and the SGD optimization function at a learning rate of 0.001.

4.2 Hardware Evaluation

Among the various available tools for implementing hardware designs of CNNs on different FPGAs, Xilinx Vivado ® High-Level-Synthesis (Vivado_HLS) is commonly used in literature for the sake of productivity at the cost of hardware efficiency and performance [9], [32]–[36]. Hence, we leverage the Vivado_HLS and Vivado_IDE (v2018.3) tools to realize hardware circuits. The FPGA synthesis is executed with chip ZC7Z020-1CLG484C for the property with benchmarks in comparison. We use Vivado_HLS to compare hardware resources and latency of the IP core between the original approach and the proposal with a 32-bits floating-point and 16-bits fixed point at a frequency of 100 Mhz. We conduct our IP core into an embedded system to verify area and power on real FPGA at 115MHZ of the frequency with 24-bits fixed point.

First, we evaluate the SLIT layer in two cases with eight binary output feature maps and eleven output channels that concatenate eight binary channels with three input channels. Second, we stack another CONV layer after the first CONV layer to assess how to compose the SCONV in the proposed topology. We have two CONV layers in

the primary, but in the proposal, we concatenate SLIT and SCONV layers. Third, the MP proposal on CNNs is analyzed by appending one MP after the first CONV layer. We explain two cases: the first case is the structure having CONV, MP layers and the second is CONV, MP, CONV layers in the model. We handle the SLIT, SMP layers, and the SLIT, SMP, SCONV layers to compare with the data obtained from the conventional scheme. Next, the FC proposal is studied by linking the SLIT, SMP, and SFC layers. Finally, we investigate the architecture of the Lenet-5 and VGG-16 models as the analyzed standard of the proposed networks on hardware to compare with state-of-the-art. How to replace the first three layers is showed in the Lenet-5, and VGG-16 represents how to reconstruct the first two layers on the deep neural networks.

5. Experimental Results

5.1 Software Performance Analysis

Figure 9 shows the accuracy of the MNIST, CIFAR-10, CIFAR-100, SVHN and ImageNet datasets. A small decrease in accuracy of 0.27% from 99.07% to 98.8% with the MNIST database has been observed when being compared between the Lenet-5 proposal and the conventional Lenet-5 paradigm. Moreover, it slightly decreases from 0.5% to 1.5% with CIFAR-10 and CIFAR-100 datasets, and around 2.2% on the ImageNet. It also remarks efficiently on the small CNN model that is experimented with the SVHN dataset. With complicated models such as VGG-16 and VGG-19, the loss of accuracy ranges from 0.5% to 2.2%. A training time reduction shown in Fig. 10 compensates for the loss of accuracy when our proposal is applied. Total training time is diminished by 40%, 40%, and 32%, corresponding with MNIST, CIFAR, and SVHN databases on Lenet-5 and CNN models. It also decreases by approximately 10% on larger paradigms such as VGG-16 and VGG-19 with the CIFAR database. Because the model verified with the VGG-16 on ImageNet takes a long time for training one epoch, this case is not revealed in Fig. 10.

Ordinarily, the first and second CONV layers contribute 92.4% of MAC operations, while the FC layers offer 7.6% of MAC operations on the Lenet-5 model. Parameter and operation reduction highlight the contribution of the proposed model for the training phase on CNNs. Results indicate a considerable efficiency obtained on the proposed Lenet-5 model. Table 1 and Table 2 show that a total of 1×5×5×6 = 150 parameters and 463K MAC operations are pruned in the proposal when evaluated with the MNIST database. Remarkably, with the CIFAR dataset that has three channels, 3×5×5×8 = 450 parameters and a total of 3×5×5×32×32×6 + 6×5×5×16×16×16 = 1.07G MAC operations are excluded. The proposal decreases approximately by 90% MAC operations and leads to training time reduction during the training phase on the Lenet-5 model. An entirety of 1728 parameters and 1.77M MAC operations are also eliminated in the first layer on the VGG-16 model. In short, to compare with the original approach and Ref. [9], our model illustrates better MAC operation optimization on

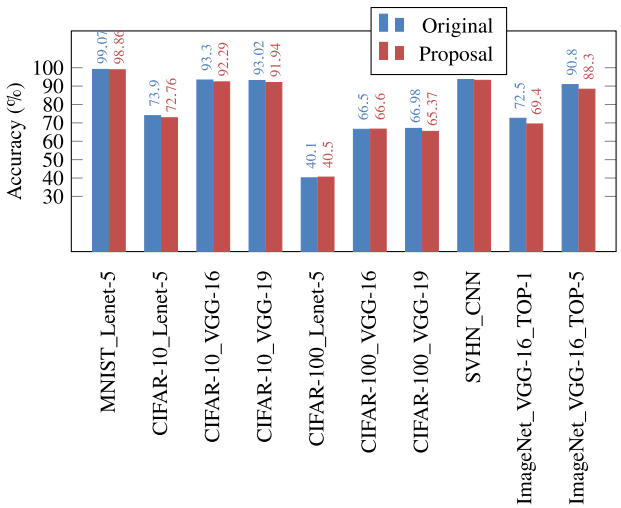

Fig. 9 Comparing accuracy between the original model and the proposed model

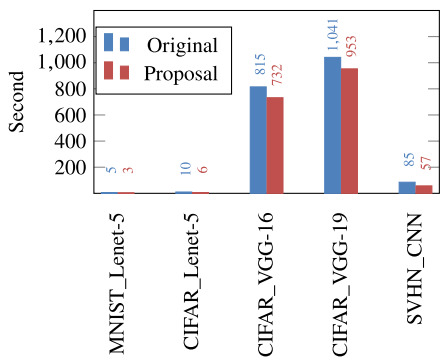

Fig. 10 Comparing training time of one epoch between the original model and the proposed model

Table 1 Comparison parameters on Lenet-5 and VGG-16 model

Layer	Kernal	Original	Optimized [9]	Proposal
Lenet-5 model				
CONV1	1×5×5×6	150	336	0
CONV2	6×5×5×16	2400	2752	2400
VGG-16 model				
CONV1	3×3×3×64	1728	41K	0
CONV2	64×3×3×64	2400	49K	2400

Table 2 Comparison operations on Lenet-5 and VGG-16 model

Layer	Layer	Original	Optimized [9]	Proposal
Lenet-5 model				
CONV1	1×28×28	117.6K (a)	225K (a)	0
MP	6×24×24	27.6K (b)	27.6K (b)	9216 (c)
CONV2	6×12×12	345.6K (a)	419.5K (a)	345.6K (d)
VGG-16 model				
CONV1	3×32×32	1.77M (a)	37.6G (a)	0
CONV2	64×32×32	37.7M (a)	50.3G (a)	37.7M (d)

a: MAC, b: Comparator, c: OR, d: Multiplexer

the Lenet-5 and VGG-16 models.

5.2 Hardware Performance Analysis

Table 3 exposes the reduction of hardware resources and performance for the first layer. The SLIT layer employs fewer LUTs, FFs, BRAM and DSP48E blocks than the CONV layer. Especially, the DSP48E blocks are humbled eight times in the case of SLIT(8). Latency achieves a 13.47/0.427 = 31.5× reduction compared with the CONV(8) layer and a factor of 38× decrease with the CONV(11) layer. Table 4 (a) reveals the hardware resources and latency for the second layer. By replacing all MAC operations with the MUX function in case SLIT(8) + SCONV(16), hardware utilization is notably reduced. For example, 2 DSP48E blocks are proportional to 13 DSP48E blocks in the standard design. BRAM blocks are lessened 18/2 = 9× between two models. Latency is also decreased remarkably in our proposal when being compared with the traditional topology.

Table 4 (b) reveals the max-pooling layer performance. By replacing three comparator operations with an OR gate, latency or speed is reduced from 13.6 ms to 0.46 ms. Hardware resources that estimate the IP core area extremely decrease on DSP48E and BRAM blocks. A factor of approximately 92% hardware resource reduction is observed when our SMP layer is compared to the second traditional MP layer. In a more complicated case like SLIT(11) + SMP(2) + SCONV(16), our reconfiguration not only reduces significant hardware resources but also demand 53.6 ms, a reduction from as 96.9 ms as in the case of CONV(11) + MP(2) + CONV(16). The FC proposal is analyzed by combining SLIT, SMP, and SFC layers. By replacing the multiplication matrix with MUX functions, Table 4 (c) proves that our suggestion also works better than the traditional process in terms of hardware resource utilization and execution time requirements. In short, four loops in the SLIT layer, OR gate in the SMP layer, and MUX operation in the SCONV layer result in enormous hardware resources and latency reduction.

Table 3 Comparing hardware resources and latency for the first layer

Layers	CONV(8)	SLIT(8)	CONV(11)	SLIT(11)
Floating-point	**32-bits**	**32-bits**	**32-bits**	**32-bits**
LUT	724	241	815	549
FF	960	233	1073	657
DSP48E	8	0	8	3
BRAM	2	1	8	1
Latency (ms)	13.47	0.427	40.17	1.04

5.3 Design Comparison

As a comparison with the traditional CNNs on Lenet-5 and VGG models, our scheme replaces the first three layers on the conventional Lenet-5 model and the first two layers on VGG. By synthesizing with the Vivado_HLS tool, Table 5 (a) shows the proposal has consumed less than 52.9% BRAM and 33.3% DSP48E blocks compared with the traditional Lenet-5 model. Moreover, our scheme achieves about 1- 20.78/34.3 = 0.394 or 39% latency reduction without using optimized methods such as #parama HLS_PIPElINE or #parama HLS_UNROLL. We have also investigated hardware resources and latency for the VGG-16 scheme. Table 5 (b) demonstrates an efficient latency reduction on the first and second layers. The proposal requires 1.04 ms as comparing 249 ms on the first layer and a 6× latency reduction in the second layer. The hardware resources also

Table 4 Comparing hardware resource utilization and latency between the traditional approach and the proposal

(a) Comparing hardware resources and latency for the proposed second layer				
Layers	CONV(8)+CONV(16)	SLIT8+SCONV(16)	CONV(11)+CONV(16)	SLIT(11)+SCONV(16)
Floating-point	32-bits	32-bits	32-bits	32-bits
LUT	1303	736	1425	1223
FF	1649	769	1811	1399
DSP48E	13	2	13	8
BRAM	18	2	40	10
Latency (ms)	160.7	96.5	266.3	210.9
(b) Comparing hardware resources and latency for the proposed max pooling layer				
Layers	CONV(8)+MP(2)	SLIT(8)+SMP(2)	CONV(11)+MP(2)+CONV(16)	SLIT(11)+SMP(2)+SCONV(16)
Floating-point	32-bits	32-bits	32-bits	32-bits
LUT	1028	341	1686	1497
FF	1255	334	2071	1673
DSP48E	8	0	13	8
BRAM	18	2	48	13
Latency (ms)	13.6	0.46	96.9	53.6
(c) Comparing hardware resources and latency for the proposed fully connected layer				
Layers	CONV(8)+MP(2)+FC(512)	SLIT(8)+SMP(2)+FC(512)	CONV(11)+MP(2)+FC(1024)	SLIT(11)+SMP(2)+FC(1024)
Floating-point	32-bits	32-bits	32-bits	32-bits
LUT	1531	741	1549	1359
FF	1924	829	2006	1578
DSP48E	13	2	13	8
BRAM	22	3	48	13
Latency (ms)	105.5	60.5	454.3	379.9

Table 5 Comparing hardware resource utilization and latency on Lenet-5 and VGG-16 models between the traditional CNN approach and the proposal

(a) Lenet-5 model on MNIST database with 32-bits floating-point											
Layer	**Dimensions**	**Traditional Lenet-5**					**Proposal Lenet-5**				
		LUT	FF	BRAM	DSP48E	Latency (ms)	LUT	FF	BRAM	DSP48E	Latency (ms)
CONV1	1×28×28	723	954	2	8	10.1	241	233	1	0	0.42
MP1	6×24×24	1017	1240	10	8	10.2	341	334	2	0	0.46
CONV2	6×12×12	1618	1960	12	13	26.3	1030	1130	3	4	17.9
Lenet-5 model		4568	4371	17	24	34.3	3854	3405	8	16	20.78
(b) VGG-16 model on CIFAR-10 database with 32-bits floating-point											
Layer	**Dimensions**	**Traditional VGG-16**					**Proposal VGG-16**				
		LUT	FF	BRAM	DSP48E	Latency (ms)	LUT	FF	BRAM	DSP48E	Latency (ms)
CONV1	3×32×32	832	1106	8	8	249.48	549	657	1	3	1.04
CONV2	64×32×32	2236	2236	136	10	5246.2	1869	4573	10	8	839.6
VGG-16 model		43442	11276	480	62	49820.9	43140	10852	354	57	44503.9

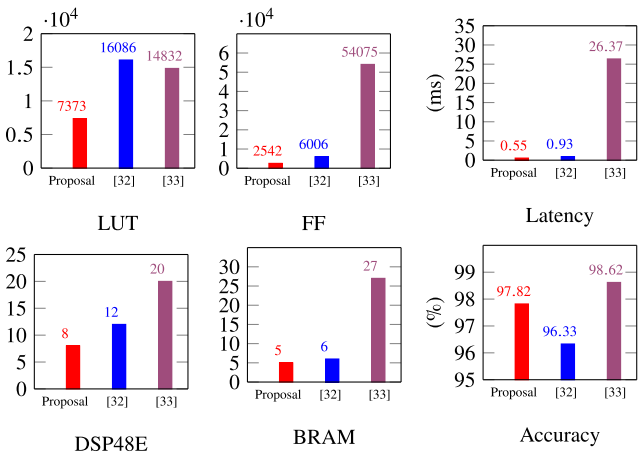

Fig. 11 Comparison hardware resources and latency of our IP core proposal with other works on Lenet-5 model at 100 MHz using Vivado_HLS tool

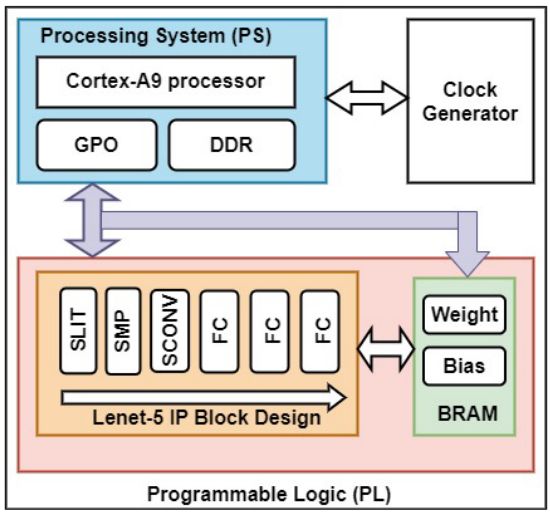

Fig. 12 The system on chip implementation of the Lenet-5 model on zynq7020 FPGA

degrade 26% in BRAM blocks and 8% in DSP28E blocks for the complete proposed VGG-16 design.

For the IP core comparison between the proposal and existing state-of-the-art using the MNIST database, the network combining CONV(8) + MP(2) + CONV(8) + MP(2) + FC(10) is proved. The proposed model consists of SLIT(8) + SMP(2) + SCONV(8) + MP(2) + FC(10). In addition to constructing the first three layers, we also use #parama HLS_PIPELINE and #parama HLS_UNROLL technologies to improve the hardware design performance. We utilize a 16-bits fixed point while still maintaining accuracy. Figure 11 reveals that the proposal demands a smaller number of hardware resources than previous works at higher accuracy. Especially, the latency achieves a 40.8% reduction over the work [32], and a factor of 26.3/0.55 = 47× decrease compared with the result reported in the previous study [33]. Besides, the hardware resource is lower with 1 BRAM block, 4 DSP48E blocks, 6006/2542 = 2.3× FFs, and 16086/7373 = 2.18× LUTs as compared with the highest current performance [32]. Moreover, our proposal maintains 97.82% accuracy higher than 96.33% in Ref. [32].

As shown in Fig. 12, the CNN accelerator design includes ARM, AXI, BRAM, and our IP core. The IP core is called in an ARM CPU-based embedded system to analyze

Table 6 Comparing resource utilization and power consumption on chip zynq7020 FPGA for Lenet-5 model

Parameter	**[34]** 24-bits fixed point	**[35]** 32-bits floating-point	**[36]** 8-bits fixed point	**Proposal** 24-bits fixed point
Frequency	166 MHZ	100 MHZ	100 MHZ	115 MHZ
LUT	38836	14659	39898	6853
FF	23408	14172	25161	6378
DSP48E	95	125	0	16
BRAM	92	119.5	24	127
Power (W)	3.32	1.8	1.758	0.456

the effectiveness of the proposed optimization technique. Table 6 exposes the comparison between our model and the previous works in hardware resources used to estimate area and power consumption. Due to the binary calculation on SLIT, SMP, and SCONV layers, the DSP48E blocks are extremely reduced in our proposal. To fairly assess, we convert 16 DSP48E blocks into 1003 LUTs, and 537 FFs in the way Ref. [36] measurement and estimate equivalently one BRAM into 256 LUTs as Refs. [43], [44]. As a result, our topology utilizes the same LUTs with a 72.5% reduction in FFs compared with the Ref. [36]. Moreover, the

proposal also employs a 0.456 W power consumption lower than works [34]–[36].

6. Conclusions

In this paper, we have created a layer that imitates the primary visual cortex principle and replaced the first few layers of conventional CNNs successfully. The innovative reconfiguration for the Lenet-5 scheme has achieved a 70% discount in hardware resources and an improvement of 39% latency at a power consumption of 0.456 W for the inference phase on FPGA. The entire convolution operations in the first two convolutional layers of the traditional CNN models are removed efficiently. Accuracy of the proposal has just slightly reduced a factor of 0.27% on Lenet-5 with MNIST dataset, approximately 1.5% on VGG-16 and VGG-19 with CIFAR dataset, 2.2% on VGG-16 with ImageNet database, and remained the same with the SVHN database. Our method is elastic to concatenate with various conventional models at high efficient energy and minimum hardware resources on FPGA. Hence, it gives a new inspiration toward combining our proposal with BinaryConnect or SqueezeNet method to obtain higher hardware design optimization. In the future, we plan to study a more extensive and scalable CNN accelerator that will integrate our proposal with other optimization approaches. We hope that SLIT can be a potential method in exploring the broad range of CNN architecture reconfiguration.

Acknowledgments

A part of this research is based on Grant-in-Aid for Scientific Research (A) JP17H00730.

References

[1] C. Cao, B. Wang, W. Zhang, X. Zeng, X. Yan, Z. Feng, Y. Liu, and Z. Wu, "An improved faster r-cnn for small object detection," IEEE Access, vol.7, pp.106838–106846, 2019.

[2] X. Lei, H. Pan, and X. Huang, "A dilated cnn model for image classification," IEEE Access, vol.7, pp.124087–124095, 2019.

[3] I. Gavrilut, A. Gacsadi, C. Grava, and V. Tiponut, "Vision based algorithm for path planning of a mobile robot by using cellular neural networks," 2006 IEEE International Conference on Automation, Quality and Testing, Robotics, pp.306–311, IEEE, 2006.

[4] J. Shang, L. Qian, Z. Zhang, L. Xue, and H. Liu, "Lacs: A high-computational-efficiency accelerator for cnns," IEEE Access, vol.8, pp.6045–6059, 2019.

[5] R. Wang, Z. Cao, X. Wang, Z. Liu, and X. Zhu, "Human pose estimation with deeply learned multi-scale compositional models," IEEE Access, vol.7, pp.71158–71166, 2019.

[6] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.5687–5695, 2017.

[7] Y.-H. Chen, T. Krishna, J.S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE journal of solid-state circuits, vol.52, no.1, pp.127–138, 2016.

[8] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," arXiv preprint arXiv:1602.07360, pp.1–13, 2016.

[9] M. Hailesellasie, S.R. Hasan, F. Khalid, F.A. Wad, and M. Shafique, "Fpga-based convolutional neural network architecture with reduced parameter requirements," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp.1–5, IEEE, 2018.

[10] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014, pp.1269–1277, Neural information processing systems foundation, 2014.

[11] Y. Umuroglu, N.J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.65–74, 2017.

[12] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable fpgas," Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.15–24, 2017.

[13] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," European conference on computer vision, pp.525–542, Springer, 2016.

[14] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.5918–5926, 2017.

[15] J. Chung, W. Choi, J. Park, and S. Ghosh, "Domain wall memory-based design of deep neural network convolutional layers," IEEE Access, vol.8, pp.19783–19798, 2020.

[16] D. Jung, S. Lee, W. Rhee, and J.H. Ahn, "Partitioning compute units in cnn acceleration for statistical memory traffic shaping," IEEE Computer Architecture Letters, vol.17, no.1, pp.72–75, 2017.

[17] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, "A lightweight privacy-preserving cnn feature extraction framework for mobile sensing," IEEE Transactions on Dependable and Secure Computing, pp.1–15, 2019.

[18] A. Ferdowsi, U. Challita, and W. Saad, "Deep learning for reliable mobile edge analytics in intelligent transportation systems: An overview," IEEE vehicular technology magazine, vol.14, no.1, pp.62–70, 2019.

[19] F.U.D. Farrukh, T. Xie, C. Zhang, and Z. Wang, "Optimization for efficient hardware implementation of cnn on fpga," 2018 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA), pp.88–89, IEEE, 2018.

[20] S. Li, W. Wen, Y. Wang, S. Han, Y. Chen, and H. Li, "An fpga design framework for cnn sparsification and acceleration," 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp.28–28, IEEE, 2017.

[21] H. Ando, Y. Niitsu, M. Hirasawa, H. Teduka, and M. Yajima, "Improvements of classification accuracy of film defects by using gpu-accelerated image processing and machine learning frameworks," 2016 Nicograph International (NicoInt), pp.83–87, IEEE, 2016.

[22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," Proceedings of the 22nd ACM international conference on Multimedia, pp.675–678, 2014.

[23] S. Mittal, "A survey of fpga-based accelerators for convolutional neural networks," Neural computing and applications, pp.1–31, 2018.

[24] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded fpga platform for convolutional neural network," Proceedings of the 2016 ACM/SIGDA International Symposium on Field-

Programmable Gate Arrays, pp.26–35, 2016.
[25] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.161–170, 2015.
[26] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, "Optimizing loop operation and dataflow in fpga acceleration of deep convolutional neural networks," Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.45–54, 2017.
[27] J. Iwamoto, Y. Kikutani, R. Zhang, and Y. Nakashima, "Daisy-chained systolic array and reconfigurable memory space for narrow memory bandwidth," IEICE Transactions on Information and Systems, vol.E103-D, no.3, pp.578–589, 2020.
[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol.86, no.11, pp.2278–2324, 1998.
[29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng, "Reading digits in natural images with unsupervised feature learning," NIPS Workshop on Deep Learning and Unsupervised Feature Learning, pp.1–9, 2011.
[30] A. Krizhevsky, G. Hinton, et al., "Learning multiple layers of features from tiny images," Technical report, University of Toronto, pp.32–33, 2009.
[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," International Journal of Computer Vision, vol.115, no.3, pp.211–252, April 2015.
[32] T.-H. Tsai, Y.-C. Ho, and M.-H. Sheu, "Implementation of fpga-based accelerator for deep neural networks," 2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp.1–4, IEEE, 2019.
[33] S. Ghaffari and S. Sharifian, "Fpga-based convolutional neural network accelerator design using high level synthesize," 2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS), pp.1–6, IEEE, 2016.
[34] G. Feng, Z. Hu, S. Chen, and F. Wu, "Energy-efficient and high-throughput fpga-based accelerator for convolutional neural networks," 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), pp.624–626, IEEE, 2016.
[35] D. Rongshi and T. Yongming, "Accelerator implementation of lenet-5 convolution neural network based on fpga with hls," 2019 3rd International Conference on Circuits, System and Simulation (ICCSS), pp.64–67, IEEE, 2019.
[36] M. Zhao, X. Li, S. Zhu, and L. Zhou, "A method for accelerating convolutional neural networks based on fpga," 2019 4th International Conference on Communication and Information Systems (ICCIS), pp.241–246, IEEE, 2019.
[37] G.N. Reeke Jr and O. Sporns, "Behaviorally based modeling and computational approaches to neuroscience," Annual Review of Neuroscience, vol.16, no.1, pp.597–623, 1993.
[38] D.D. Cox and T. Dean, "Neural networks and neuroscience-inspired computer vision," Current Biology, vol.24, no.18, pp.R921–R929, 2014.
[39] C.D. James, J.B. Aimone, N.E. Miner, C.M. Vineyard, F.H. Rothganger, K.D. Carlson, S.A. Mulder, T.J. Draelos, A. Faust, M.J. Marinella, J.H. Naegle, and S.J. Plimpton, "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications," Biologically Inspired Cognitive Architectures, vol.19, pp.49–64, 2017.
[40] G. Leuba and R. Kraftsik, "Changes in volume, surface estimate, three-dimensional shape and total number of neurons of the human primary visual cortex from midgestation until old age," Anatomy and embryology, vol.190, no.4, pp.351–366, 1994.
[41] C. Lv, Y. Xu, X. Zhang, S. Ma, S. Li, P. Xin, M. Zhu, and H. Ma, "Feature extraction inspired by v1 in visual cortex," Ninth International Conference on Graphic and Image Processing (ICGIP 2017), p.106155C, International Society for Optics and Photonics, 2018.
[42] T.D. Tran, M. Kimura, and Y. Nakashima, "Primary visual cortex inspired feature extraction hardware model," 2020 4th International Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom), pp.20–24, IEEE, 2020.
[43] G.P. Saggese, A. Mazzeo, N. Mazzocca, and A.G. Strollo, "An fpga-based performance analysis of the unrolling, tiling, and pipelining of the aes algorithm," International Conference on Field Programmable Logic and Applications, pp.292–302, Springer, 2003.
[44] L. Li, S. Lin, S. Shen, K. Wu, X. Li, and Y. Chen, "High-throughput and area-efficient fully-pipelined hashing cores using bram in fpga," Microprocessors and Microsystems, vol.67, pp.82–92, 2019.

Thi Diem Tran received her Bachelor and Master degrees in physical electronics from University of Science, Vietnam National University - Ho Chi Minh (VNU-HCM) in 2006 and 2009, respectively. She is currently working toward the Ph.D. degree from the Nara Institute of Science and Technology (NAIST), Japan. Her research interests include machine learning and image processing.

Yasuhiko Nakashima received B.E., M.E., and Ph.D. degrees in Computer Engineering from Kyoto University in 1986, 1988 and 1998, respectively. He was a computer architect in the Computer and System Architecture Department, FUJITSU Limited from 1988 to 1999. From 1999 to 2005, he was an associate professor at the Graduate School of Economics, Kyoto University. Since 2006, he has been a professor in the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include computer architecture, emulation, circuit design, and accelerators. He is a member of IEEE CS, ACM, and IPSJ.