IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 1, 2021, accepted July 18, 2021, date of publication July 26, 2021, date of current version July 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3099075

A Resource Efficient Integer-Arithmetic-Only FPGA-Based CNN Accelerator for Real-Time Facial Emotion Recognition

JAEMYUNG KIM[1], JIN-KU KANG[1], (Senior Member, IEEE), AND YONGWOO KIM[2], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, Inha University, Incheon 22212, South Korea
[2]Department of System Semiconductor Engineering, Sangmyung University, Cheonan 31066, South Korea

Corresponding authors: Yongwoo Kim (yongwoo.kim@smu.ac.kr) and Jin-Ku Kang (jkang@inha.ac.kr)

ABSTRACT Recently, many researches have been conducted on recognition of facial emotion using convolutional neural networks (CNNs), which show excellent performance in computer vision. To obtain a high classification accuracy, a CNN architecture with many parameters and high computational complexity is required. However, this is not suitable for embedded systems where hardware resources are limited. In this paper, we present a lightweight CNN architecture optimized for embedded systems. The proposed CNN architecture has a small memory footprint and low computational complexity. Furthermore, a novel hardware-friendly quantization method that uses only integer-arithmetic is proposed. The proposed hardware-friendly quantization method maps the scale factors to power-of-two terms and replaces multiplication and division operations using scale factors with shift operations. To improve the generalization and classification performance of the CNN, we create the FERPlus-A dataset. This is a new training dataset created using a variety of image processing algorithms. After training with FERPlus-A, quantization has been performed. The size of a quantized CNN parameter is about 0.39 MB, and the number of operations is about 28 M integer operations (IOPs). By evaluating the performance of the quantized CNN that uses only integer-arithmetic on the FERPlus test dataset, the classification accuracy is approximately 86.58%. It achieved higher accuracy than other lightweight CNNs in prior studies. The proposed CNN architecture that uses only integer-arithmetic is implemented on the Xilinx ZC706 SoC platform for real-time facial emotion recognition by applying parallelism strategies and efficient data caching strategies. The FPGA-based CNN accelerator implemented for real-time facial emotion recognition achieves about 10 frame per second (FPS) at 250 MHz and consumes 2.3 W.

INDEX TERMS Emotion recognition, convolutional neural network, quantization, accelerator, FPGA.

I. INTRODUCTION

Today, computers play a central role in industry and society, and are rapidly becoming a part of everyday life. Accordingly, the need for research on the interaction between humans and computers is increasing. For smooth interaction between them, a computer must be able to analyze human intention and respond accordingly. Emotions that appear in facial expressions are a universal and effective way of expressing human intentions. Analyzing human intentions through the emotions revealed in faces is called facial emotion recognition technology, and is used in diverse fields such as automobiles and robot industries. To understand accurately the emotions displayed on human faces, a computer must recognize the face and automatically classify the emotions according to specific emotion groups.

Ekman *et al.* [1] defined seven basic emotions: happiness, anger, fear, surprise, disgust, sadness, and neutral. Moreover, they found that basic emotions are perceived in the same way regardless of human culture. In the field of computer vision, the Facial Action Coding System (FACS), a facial expression analysis method proposed in [2], was used as a typical facial emotion recognition model. FACS defines the basic movement of the facial muscles as Action

The associate editor coordinating the review of this manuscript and approving it for publication was Gulistan Raja.

Units (AUs), and then uses combinations of AUs to recognize facial emotions.

However, the development of a real-time automation system has thus far failed due to inadequate computational capabilities and inefficient pre-processing algorithms.

Due to the advancement of image processing algorithms and the improvement of computing power, facial emotion recognition technology has evolved into a method of extracting hand-crafted features through a three-phase pipeline [3]. This approach consists of image pre-processing, feature extraction, and expression classification phases. The image pre-processing phase removes irrelevant information using filtering and histogram equalization, and reinforces information related to facial emotions. The feature extraction phase extracts face features from images using a feature extractor such as the Gabor Wavelet Kernel [4], Local Binary Pattern (LBP) [5], Active Shape Model (ASM) [6], and the Harr-Like Feature Template [7]. The expression classification phase classifies the extracted features into emotion groups using a classifier such as a support vector machine (SVM), k-nearest neighbor (KNN), or AdaBoost [8]. The method using hand-crafted features requires designing an appropriate feature extractor and expression classifier individually, so both phases cannot be optimized at the same time. Moreover, external factors (e.g., pose-variant, occlusion, and illumination) may cause severe performance degradation.

Due to recent breakthroughs in hardware technology and big data technology, a deep learning-based method has been applied in various applications because it shows excellent performance for complicated and complex problems. Especially in computer vision, such as image classification and object detection, the method using convolutional neural networks (CNNs) has shown outstanding performance. Unlike conventional methods, the CNN-based approach is more robust in environments with noise, significantly reducing reliance on image pre-processing and feature extraction. In addition, this method can optimize the parameters at once via end-to-end training. In the study of facial emotion recognition, the CNN-based method showed higher performance than the conventional method, but many convolution layers were used when constructing the network. A network such as this requires many operations and memory footprints. Therefore, it is limited when used in an application that requires real-time processing, such as an advanced driver assistance system (ADAS) [9]. Moreover, it is not suitable for environments with limited hardware resources, such as virtual reality (VR) [10].

In this paper, we propose an FPGA-based CNN accelerator for real-time facial emotion recognition. The accelerator is optimized for embedded systems and applied a novel hardware-friendly quantization method. The main contributions of this work are as follows:

- To improve facial emotion recognition performance, FERPlus-A is presented. This is a new training dataset created by applying various image processing algorithms such as bilateral filtering [11], contrast limited adaptive history equalization (CLAHE) [12], and edge enhancement.
- A lightweight CNN architecture is proposed for embedded devices. The proposed CNN is optimized the number of operations and memory footprint using the modified fire module in SqueezeNet [13] as the basic computation module. The proposed CNN showed approximately 86.58% facial emotion recognition accuracy when evaluated using the FERPlus test dataset. This is the highest performance reported, compared with previous works using a lightweight CNN.
- Log level threshold quantization (LLTQ) is proposed, which is a novel hardware-friendly quantization method. The proposed method can replace both multiplication and division operations using scale factors with shift operations. The proposed quantized CNN parameter size is approximately 0.39 MB, and the number of operations is about 28 M integer operations (IOPs). In addition, it was confirmed that the performance of CNN that uses only 8-bit integer-arithmetic exhibited no drop in accuracy compared to using a 32-bit floating-point.
- The proposed CNN optimized for embedded devices is implemented on a Xilinx ZC706 evaluation board by applying parallelism strategies and efficient data caching strategies for real-time facial emotion recognition. The FPGA-based CNN accelerator, which uses only 8-bit integer-arithmetic, used 5,090 LUTs, 7,588 Flip-Flops, 61 BRAMs, and 49 DSPs. The accelerator achieved approximately a frame rate of 10 frame per second (FPS) while consuming 2.3 W of power.

The rest of this paper is organized as follows. In Section II, related works on facial emotion recognition using a lightweight CNN, quantization, and FPGA-based CNN accelerator for facial emotion recognition are introduced and analyzed. Section III describes the proposed design flow, a lightweight CNN architecture optimized for embedded devices, a novel hardware-friendly quantization method that reduces the disadvantages of existing quantization work, the integer scale conversion method, and the FPGA-based CNN accelerator for real-time facial emotion recognition. In Section IV, the experimental results and performance of this paper are reported and compared with those in previous works. Finally, Section V provides the conclusions of this paper.

II. RELATED WORKS

In this section, related works (facial emotion recognition using lightweight CNN, quantization, and FPGA-based CNN accelerator for facial emotion recognition) are introduced and analyzed.

A. FACIAL EMOTION RECOGNITION USING A LIGHTWEIGHT CNN

Several facial emotion recognition studies using a lightweight CNN have been reported [14]–[23]. Barsoum *et al.* [14]

constructed the FERPlus dataset, which addressed the problem of the FER2013 [24] dataset using the crowd-sourcing and label distribution methods, and four different training schemes using a lighten VGG [25] were proposed. Li *et al.* [15] extracted identity features and emotion features through DeepID2 [26] and ResNet [27], respectively, and then classified them by combining features from the output of each CNN. Wikanningrum *et al.* [16] proposed a transfer learning and ensemble method using pre-trained parameters with a dataset frequently used in image classification. Pandey *et al.* [17], [18] suggested an approach to train a MobileNetv2 [28] with a dataset incorporating a Sobel gradient and Laplacian of the original image dataset. Based on the method proposed in [18], Pandey *et al.* [19] used a method of training by adding center loss [29] to global loss to improve the discrimination power of the CNN.

Saurav *et al.* [20] constructed two custom CNNs using gradient weighted class activation mapping (Grad-CAM). They also proposed a dual-integrated CNN, which combines the features extracted from two CNNs and uses them for facial emotion recognition. Miao *et al.* [21] proposed SHCNN, which can alleviate the over-fitting problem for a relatively small dataset and recognize both static and micro-expressions simultaneously. Lian *et al.* [22] introduced a method to quantify the contribution of various facial areas using class activation mapping (CAM) from features extracted through a DenseNet [30]. Zhao *et al.* [23] created a new dataset integrating two similar emotion groups into one. They proposed a method to train the DenseNet by applying face detection and face alignment methods.

However, the previous works [14], [21] occupied a large amount of memory space and had a problem of high computational complexity. In earlier works [15], [20], the CNN was used with smaller parameters, but because two CNNs were used for inference, they still occupied a large amount of memory space and remain unsuitable for real-time processing. In other cases [16]–[19], CNN was used with small parameters and low computational complexity, but their facial emotion recognition accuracy is very poor. Even in works using a small number of parameters [22], [23], they were not suitable for real-time processing because they require too much memory space due to dense connectivity. This is a characteristic of the DenseNet. To address these problems, further research is needed to design a lightweight CNN architecture by optimizing memory footprint and computational complexity while maintaining the accuracy of facial emotion recognition.

B. QUANTIZATION

Quantization is a method that converts parameters trained with a 32-bit floating-point (FP) into a low-bit fixed-point (FX) or an integer (INT). Applying quantization makes it possible to optimize memory usage and to reduce the number of calculations using simpler hardware. Quantization is divided into two methods. The post-training quantization (PTQ) method [31]–[34] minimizes quantization errors by applying calibration when inferencing using pre-trained parameters. The quantization-aware training (QAT) method [35]–[39] retrains parameters considering the effects of quantization.

Nagel *et al.* [31] proposed bias correction and cross-layer equalization methods to solve biased weight errors and imbalances during the quantization process. Choukroun *et al.* [32] introduced a method of minimizing the quantization error using the least mean squared error between the floating-point value and the quantized. Banner *et al.* [33] minimized quantization errors by calculating the optimal clipping value using analytical clipping for integer quantization (ACIQ). Cai *et al.* [34] suggested a distilled dataset using the mean and variance of the batch normalization (BN) layer to obtain the optimal quantization range.

Because the PTQ method does not require a retraining process, it saves computing resources and optimization time and enables rapid deployment. However, existing researches [31], [32] need fine-tuned values to minimize loss of accuracy. In [33] and [34], different numbers of bits should be assigned for each channel to minimize the drop in accuracy. In other works [32], [33] should obtain separate scale factors for each channel to maintain accuracy. Moreover, because all PTQ methods require floating-point parameters, dedicated operators are required when implementing the hardware.

Choi *et al.* [35] proposed a method to minimize quantization errors by defining a parameterized clipping activation (PACT) function to find appropriate scale factors. Jacob *et al.* [36] defined the operation of the quantizer to perform matrix multiplication using only integer-arithmetic and presented a layer fusion method. Jung *et al.* [37] suggested a method that minimizes task loss using parameterized quantization intervals. Esser *et al.* [38] proposed a method of learning optimal quantization mapping by making scale factors learnable parameters. Jain *et al.* [39] proposed a method of mapping scale factors to power-of-two terms using scale factor conversion equation.

The QAT method can obtain higher accuracy than the PTQ method by training quantization parameters with other parameters. However, some works [37], [38] did not quantize the BN layer. In [35]–[38], the parameters required for quantization are floating-point, and dedicated operators are necessary when implementing hardware. In another approach [39], calibration of the activation scale factor is required before starting QAT. To compensate for these problems, a hardware-friendly quantization method that does not require pre-calibration step is needed.

C. FPGA-BASED CNN ACCELERATOR FOR FACIAL EMOTION RECOGNITION

Performing facial emotion recognition using a GPU provides the best performance in throughput and speed. However, it is difficult to use a GPU in embedded systems. These systems require solutions that consume less energy and use fewer hardware resources. A field programmable gate array (FPGA) has flexibility that allows designers to adapt designs

to fit an application. It is suitable for real-time processing in embedded systems because it can be programmed to enable optimal operation speed and reasonable power consumption.

Due to these advantages, several FPGA-based CNN accelerators for facial emotion recognition have been proposed [40]-[44]. Phan-Xuan *et al.* [40] implemented a CNN accelerator on a Xilinx Zynq-XC7Z020 FPGA using high-level-synthesis (HLS). They designed the CNN accelerator that stores the results and inputs of the layer in DRAM to reduce block RAM (BRAM) usage, and used VDMA for fast DRAM access. Phuc *et al.* [41] designed a CNN accelerator by applying an efficient structure and data pre-processing method for facial emotion recognition and implemented it on Altera DE-10 FPGA using Verilog HDL. In Vinhe *et al.* [42], the accelerator configured the processing engine core on Altera DE-10 FPGA to accelerate the convolution operation. Ding *et al.* [43] proposed the accelerator on Altera Cyclone-V FPGA using Verilog HDL with different parallel processing for different convolution layers constituting the CNN. In Ding *et al.* [44], the CNN accelerator implemented on Altera Cyclone-V FPGA using Verilog HDL. They designed a configurable convolution computing array to maximize DSP usage on the FPGA.

In previous works [40]–[42], a simple CNN architecture was implemented involving fully-connected, convolution, and pooling layers. Its number of parameters and computational complexity was low, but so was its accuracy, making it unsuitable for applications requiring highly accurate facial emotion recognition. In [43] and [44], DeepID architecture is implemented on an FPGA that requires a great deal of hardware resources, making it unusable in embedded systems. Therefore, it is necessary to design a low-power, low-cost FPGA-based CNN accelerator for real-time facial emotion recognition and that retains its high facial emotion recognition performance.

III. PROPOSED METHODS

A. PROPOSED DESIGN FLOW

In this paper, a novel lightweight CNN architecture is proposed that is suitable for embedded systems. This is achieved by optimizing the memory footprint and computational complexity while maintaining the accuracy of facial emotion recognition. In addition, log level threshold quantization (LLTQ) is proposed, which addresses the disadvantages of the previous works [38], [39]. Moreover, a low-power, low-cost FPGA-based CNN accelerator is proposed for real-time facial emotion recognition.

Fig. 1 shows the overall process for implementing the proposed real-time facial emotion recognition application. First, a new training dataset was generated combining various image processing algorithms to enhance CNN facial emotion recognition performance. Next, a novel lightweight CNN optimized for embedded systems as floating-point was trained. Then, LLTQ was performed, which is a hardware-friendly quantization method that uses only integer-arithmetic. Finally, a CNN accelerator IP was designed using an HLS compiler and a device driver was created using a standard C/C++ compiler. The newly developed IP and device driver were implemented in the PL and PS areas of the heterogeneous SoC platform for a real-time facial emotion recognition application.

B. PROPOSED LIGHTWEIGHT CNN ARCHITECTURE FOR EMBEDDED SYSTEMS

In this section, a lightweight CNN architecture is introduced that provides optimized memory usage and computational complexity on embedded devices. Fig. 2 shows the proposed lightweight CNN architecture consisting of 21 convolution layers and one global average pooling layer. The computation block includes two basic computation modules, FireA and FireB. The number of filters in the squeeze layer (s) and the number of filters in the expand layer (e), which are parameters of the computation block, were increased by two times in the following computation block. The proposed CNN was designed by applying the following techniques.

1) BASIC COMPUTATION MODULE

The newly designed lightweight CNN was inspired by the fire module proposed on SqueezeNet. The fire module consists of two types of layers. The squeeze layer (SQ) can reduce the number of channels of the feature map, reducing the number of operations in the expand layer that follows. The expand layer serves to expand the channel again and comprises two types of layers: a layer with a kernel size of 3 (EX3) and a layer with a kernel size of 1 (EX1). The output feature map of the SQ is divided into two paths and enters to EX1 and EX3, respectively. Furthermore, the output feature maps of EX1 and EX3 are merged through channel-wise concatenation and then enter the next layer.

The proposed lightweight CNN uses a modified fire module as a basic computation module and consists of two types (FireA and FireB). The configuration of the expand layer of FireA and FireB is the same. However, the kernel size in the SQ is set to 3 in FireA and to 1 in FireB. The reason for setting the kernel size to 3 in FireA is to minimize the accuracy drop by maintaining an appropriate reception field.

2) ALL CONVOLUTIONAL AND REGULAR NETWORK

The proposed CNN uses only the convolutional layer with kernel size of 3 or 1, so no consideration of the acceleration methods is needed for other kernel sizes when implementing in hardware. The max-pooling layer halves the resolution of the feature map and compresses the information. It can become a bottleneck when implemented in hardware because it must find the maximum value of the feature map. The proposed CNN sets the stride of FireA to 2; this halves the resolution of the feature map and replaces the max-pooling layer.

The fully-connected (FC) layer, which classifies the extracted features at the very end of the CNN, involves many parameters. The number of parameters can be reduced by

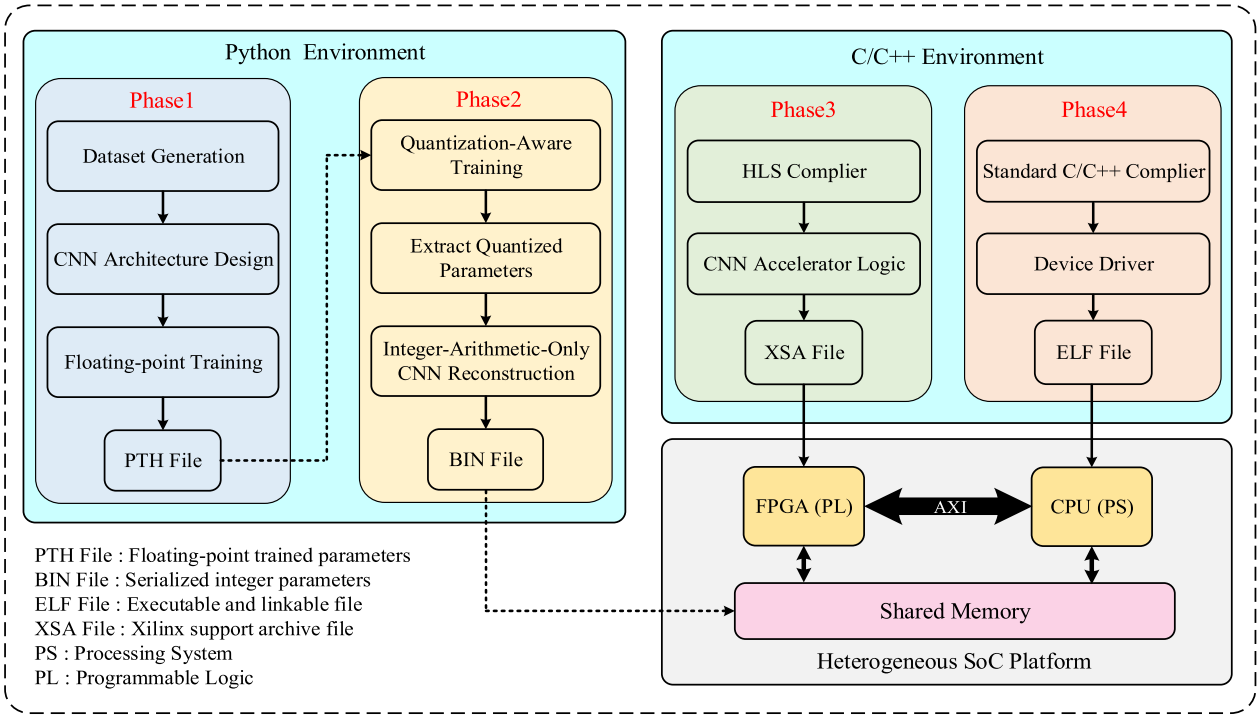

FIGURE 1. An overview of the process for implementing proposed FPGA-based CNN accelerator for real-time facial emotion recognition.

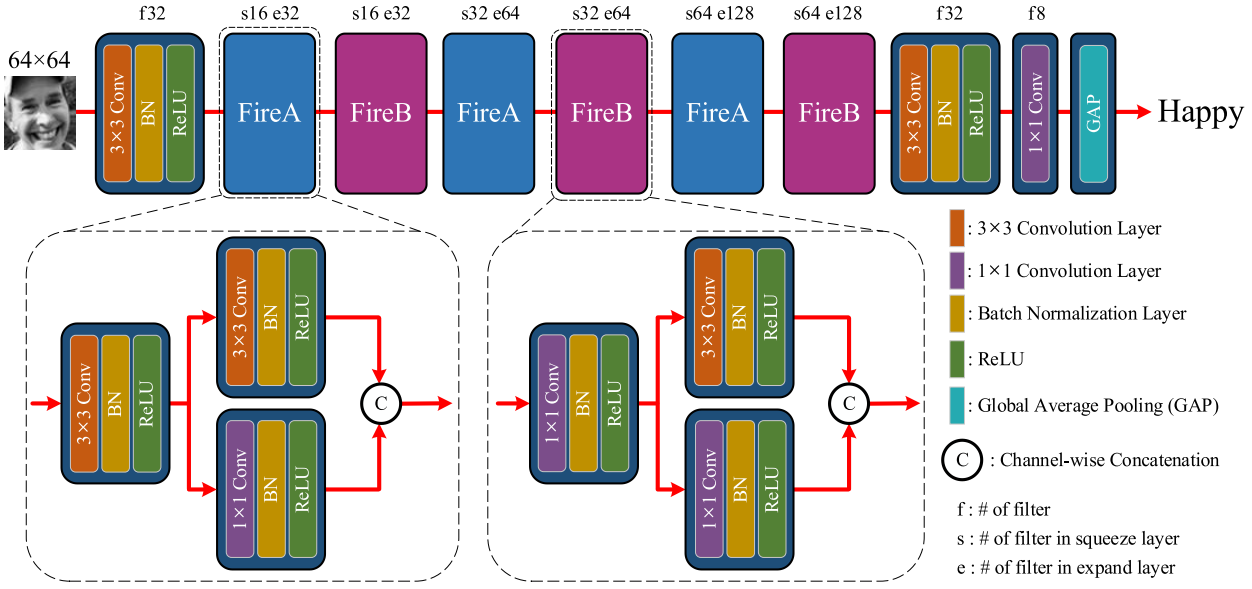

FIGURE 2. Proposed lightweight CNN architecture optimized memory usage and computational complexity.

replacing the FC layer with global average pooling and a convolution layer with a kernel size of 1. The configured classifier minimizes the number of channels to match the number of classes through the convolution layer, and then compresses the resolution of the feature map to one pixel. The activation function of all the layers is labeled ReLU. When implemented in hardware, the ReLU function is very efficient because it uses only sign-bit comparison. Using the above techniques, all layers of the CNN were made convolution layers and designed as a very regular network. In addition, a BN layer was inserted between the convolution layer and the activation function to improve the CNN convergence stability and performance. Because a BN layer can be combined with the convolution layer through BN layer fusion, the additional

number of operations and parameters generated by the BN layer can be ignored when implementing in hardware.

3) POWER-OF-2 RESOLUTION AND MULTIPLE-OF-4 OUTPUT CHANNELS

By making both the width and height of the feature map power-of-two terms, the multiplication and division operations of the part that calculates the address of the feature map can be replaced with a shift operation. In addition, the output channels of the convolution layer were all set to multiples of four for efficient parallel multiply-accumulate (MAC) hardware operations. A detailed description of the parallel MAC operations is presented in Section III-D.

C. PROPOSED QUANTIZATION METHOD

In this section, log level threshold quantization (LLTQ) is described, which is a novel hardware-friendly quantization method. Furthermore, an integer scale conversion method is described, which converts quantization-affected floating-point values to actual integer values and reconstructs CNN to use only integer-arithmetic.

1) LOG LEVEL THRESHOLD QUANTIZATION (LLTQ)

The current state-of-the-art quantization methods (learned step size quantization (LSQ) method [38] and trained quantization threshold (TQT) method [39]) use uniform quantization. A CNN with uniform quantizers inserted, performs quantization-aware training (QAT) using pre-trained floating-point parameters. Fig. 3 shows the structure of the convolution layer with a uniform quantizer inserted and the four-step operation of the quantizer. In the first step, the scale process, the quantizer maps a real value range to an integer value range of specified bit. The second step, called the round process, converts all real values mapped to an integer value range into integers. The third step is the clamp process. This process removes elements that exceed the quantization range to suit the quantization level. When the quantization bit is n, the values after the clamp process are located inside *Bound* as shown in (1):

$$Bound = \begin{cases} \left[-2^{n-1}, 2^{n-1}-1\right], & signed\ data \\ \left[0, 2^{n}-1\right], & unsigned\ data. \end{cases} \tag{1}$$

The fourth step, the de-quant process, converts values mapped to an integer range into floating-point values affected by quantization. The reason for converting the value quantized to an integer back to floating-point is that most existing deep learning frameworks using GPUs are optimized for floating-point training. The overall equation for the forward path can be described as follows:

$$x_d = clamp\left(round\left(\frac{x_f}{s}\right), Bound\right) \cdot s \tag{2}$$

where x_f is a floating-point value that is an input of the quantizer and x_d is the output value of the quantizer that passed de-quant process. The *clamp(x, Bound)* function is the second step of the uniform quantizer and cuts the rounded value (x) to fit inside the *Bound*. The *round*($\cdot$) function is the third step of the uniform quantizer. It rounds the real data to the nearest integer. A straight-through estimator (STE) [45] was used to solve the discontinuity occurring in the rounding process.

The LSQ [38] method sets the trainable parameter as a scale factor (s) and trains the scale factor along with other parameters. Therefore, the LSQ does not require scale factor conversion in the quantization process. Although it is well learned for use with the lightweight CNN described in Section III-B, there is the disadvantage that dedicated operators are needed when implementing it in hardware. This is because this is a floating-point scale factor. The TQT [39] method sets the trainable parameter as a log-threshold ($log_2 t$) and maps a scale factor to power-of-two terms as follows:

$$s = \frac{2^{ceil(log_2 t)}}{q\_level} \tag{3}$$

where the *q_level* is the value of adding 1 to the maximum value of the *Bound* in (1).

Because the TQT maps the scale factor to power-of-two terms, there is an advantage in that multiplication and division using the scale factor can be replaced with a shift operation. However, before starting retraining, calibration of the activation scale factor is required. Therefore, in this paper, a novel hardware-friendly quantization method that fixes the problems in earlier works [38], [39] is proposed. The proposed LLTQ can directly map the trainable parameter to the power-of-two scale factor. In addition, unlike TQT [39], the LLTQ method enables initializing the activation scale factor with the activation data statistics value of the first batch of the first epoch; there is no need for the pre-calibration process. Moreover, by mapping the scale factor to the power-of-two terms, the multiplication and division operations that occur in the scale and the de-quant process could all be replaced by shift operations. The performance of the LLTQ method is evaluated using only integer-arithmetic through the integer scale conversion method, which will be explained in Section III-C-2. LLTQ sets the trainable parameter as a level-threshold (*level_th*) and maps the scale factor to the power-of-two terms using the following equation:

$$s = \frac{2^{ceil(log_2(abs(level\_th * q\_level)))}}{q\_level} \tag{4}$$

where the *q_level* is the same as in (3). A detailed description of the LLTQ equation and its trainable parameter is presented in Appendix A of the supplementary material document. Many operations are required to calculate the scale factor (s). However, when performing inference with a CNN reconfigured to use only integer-arithmetic, only shift amounts of the scale factor mapped to the power-of-two terms are needed.

Additionally, several hardware-friendly quantization schemes were adapted for LLTQ. By using symmetric quantization, the computational overhead for zero-point that occurs during uniform quantization is eliminated. By introducing per-layer scaling, quantization can be performed with one

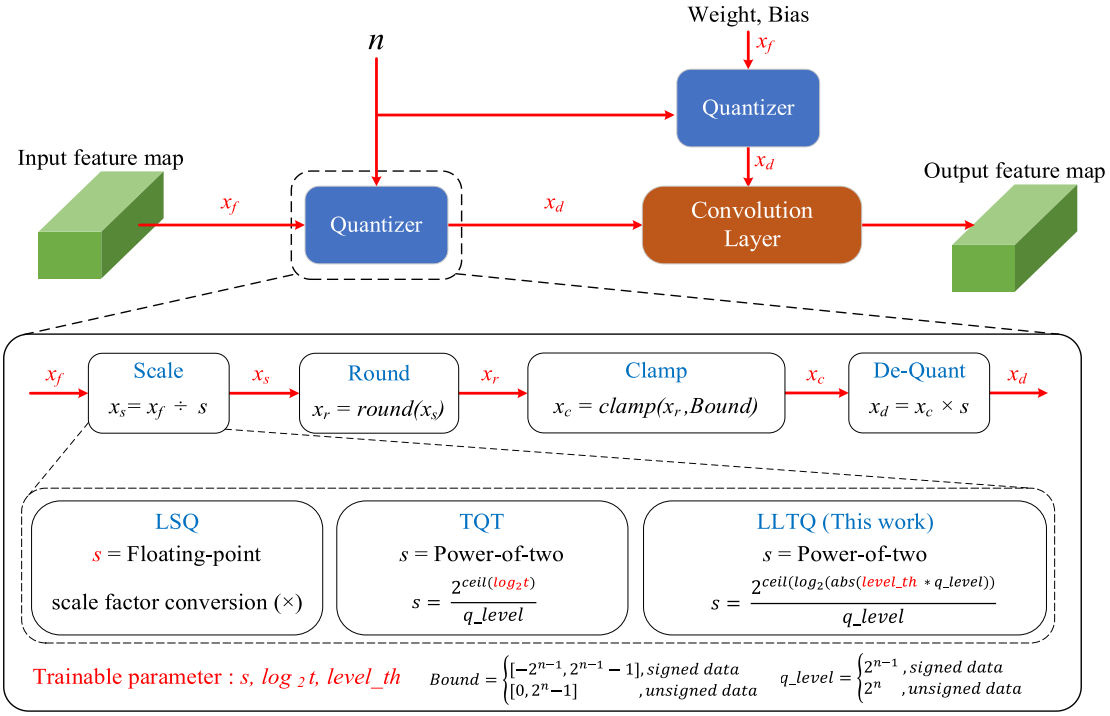

FIGURE 3. The process of a uniform quantizer using the LSQ [38], the TQT [39] method, and the proposed quantization method, LLTQ.

scale factor for all elements of a given parameter or input tensor. The BN layer fusion, which combines the parameters of the convolution layer and of the BN layer, was applied. This scheme does not need to proceed with quantization on the BN layer. This can reduce the number of operations and inference time by eliminating the multiplication and division operations required for the BN layer. The fire module is a structure in which the output of a squeeze layer enters the input of two expand layers. In floating-point, the two expand layer inputs are the same. However, in QAT, each convolution layer requires a different scale factor. Therefore, the squeeze layer has to calculate the output for each expand layer separately. This can form a bottleneck when performing CNN inference in hardware. To solve this problem, QAT was performed by making the quantizers of two expand layers into a single shared quantizer.

2) INTEGER SCALE CONVERSION

In several papers researching quantization, performance was evaluated using quantization-affected floating-point values rather than actual integer values. Using the parameters made by the proposed quantization method, the CNN was reconstructed using only integer-arithmetic through the integer scale conversion method proposed in this section. As shown in Fig. 4 (b), (d), the red line indicates the flow of floating-point data, and the black line indicates the flow of integer data. The superscript of each letter represents the type of data: f denotes floating-point data, i denotes an integer, and the subscripts k, $k+1$, $k+2$, $k+3$ below, indicate the number of each layer. The term Q_a is the integer-arithmetic-only (IAO) activation quantizer, Q_b is the IAO bias quantizer, and Q_w is the IAO weight quantizer. The terms $S_{w,k}$, $S_{w,k+1}$, and $S_{w,k+2}$ are weight scale factors, $S_{b,k}$, $S_{b,k+1}$, and $S_{b,k+2}$ are bias scale factors, and $S_{a,k}$, $S_{a,k+1}$, $S_{a,k+2}$, and $S_{a,k+3}$ are the activation scale factors. Here, *A, W, B,* and *C* marked on the dataflow arrow mean layer input, weight, bias, and activation function output, respectively. In Fig. 4, the part marked *Offline* is the process of converting quantization-affected floating-point parameters into integers. This process is done before inference and thus does not affect inference time. The process of converting quantization-affected floating-point parameters into integer parameters is represented as:

$$W_k^i = clamp(round(W_k^f \ll S_{w,k}, Bound)) \quad (5)$$

$$B_k^i = clamp(round(B_k^f \ll S_{b,k}, Bound)) \ll (S_{a,k} + S_{w,k} - S_{b,k}). \quad (6)$$

It was assumed that the input of the convolution layer is an integer quantized with the bit-precision specified at QAT. As shown in Fig. 4 (a) and (b), the general convolution layer performs MAC operation on integer scale weights and integer scale activations and adds the integer scale bias to the MAC result. As shown in (6), the bias value should then be converted to the same scale as the MAC operation result. This method is called bias up-scaling, and without this process, an error occurs because the bias and the MAC operation result have different bit-precision. After adding bias, the data passes the activation function ReLU, and the result of the ReLU can

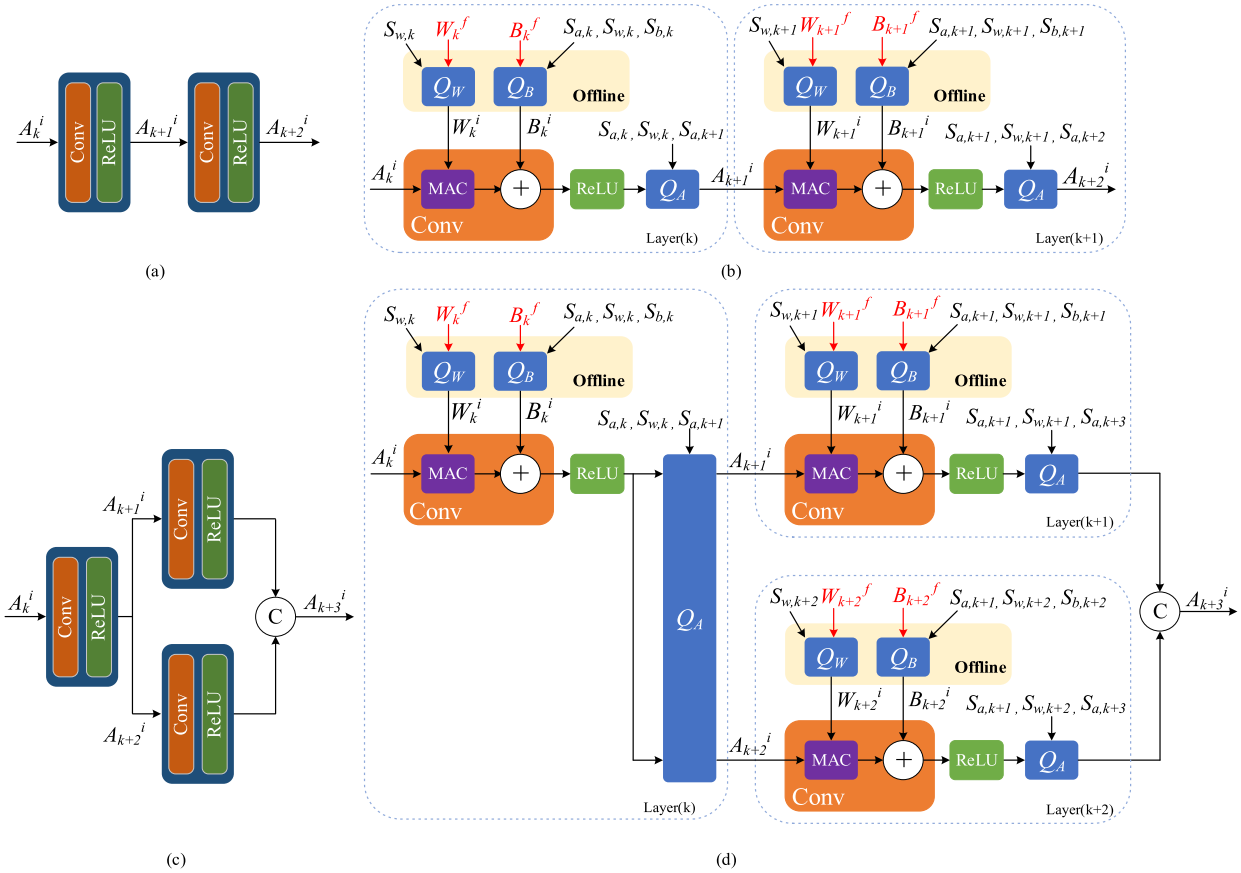

FIGURE 4. Integer scale conversion method: (a) General convolution layer. (b) Integer scale conversion method for general convolution layer. (c) Fire module. (d) Integer scale conversion method for Fire module.

be expressed as shown in (7):

$$C_k^i = ReLU(MAC\left(A_k^i, W_k^i\right) + B_k^i) \tag{7}$$

where C_k^i is the output value of ReLU, it needs to be converted from bit-precision of the current layer's MAC operation result to the bit-precision specified by QAT before entering of the next layer. This method is called activation down-scaling, and it cancels the effect on the scale factor of the current layer and scales to the input precision of the next layer. Activation down-scaling can be represented as shown in (8):

$$A_{k+1}^i = clamp(C_k^i \gg S_{a,k} + S_{w,k} - S_{a,k+1}, Bound). \tag{8}$$

In the fire module shown in Fig. 4 (c) and (d), the behavior of the squeeze layer is the same as the general convolution layer to which (5), (6), and (7) apply. Due to the introduction of a shared quantizer, a single scale factor is sufficient. Therefore, the same input can be used without calculating the input for each expand layer. The input of each expand layer can be described as in (9):

$$A_{k+1}^i, A_{k+2}^i = clamp(C_k^i \gg S_{a,k} + S_{w,k} - S_{a,k+1}, Bound). \tag{9}$$

The output of each expand layer should be channel-wise concatenate after activation down-scaling using (8). Scaling is not needed because channel-wise concatenation does not affect a value. This process is represented in (10):

$$\begin{aligned} A_{k+3}^i = concat[&clamp(C_{k+1}^i \gg S_{a,k+1} + S_{w,k+1} \\ &- S_{a,k+3}, Bound), clamp(C_{k+2}^i \\ &\gg S_{a,k+2} + S_{w,k+2} - S_{a,k+3}, Bound)]. \end{aligned} \tag{10}$$

D. PROPOSED CNN ACCELERATOR

In this section, A design methodology of a low-power, low-cost FPGA-based CNN accelerator for real-time facial emotion recognition is described. First, the entire proposed hardware architecture is presented. Then, the parallelism strategies and data caching strategies are explained, as well as the post-processing module. Last, the task-scheduling of the accelerator is described.

1) HARDWARE ARCHITECTURE

Fig. 5 shows the proposed hardware architecture of the CNN accelerator for real-time facial emotion recognition. The accelerator was implemented on a heterogeneous SoC platform composed of CPU (Processing System, PS) and FPGA (Programmable Logic, PL). The CPU allocates a memory region and transfers the kernel size, stride, padding, input

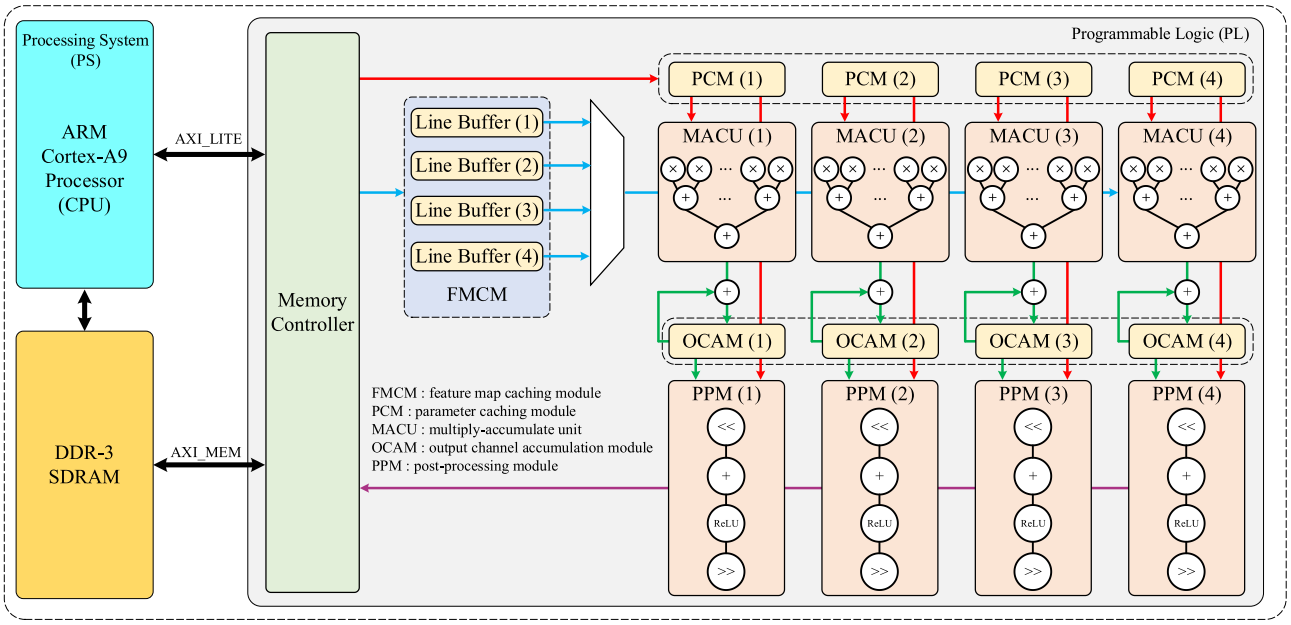

FIGURE 5. Hardware architecture of the proposed CNN accelerator for real-time facial emotion recognition.

channel, output channel, shift amounts, and address, which are each layer configuration information, to the FPGA accelerator IP through the AXI_LITE bus. Data for MAC operation, such as input images, feature maps, and parameters, are transferred from DRAM to the FPGA accelerator through the AXI_MEM bus. Each layer constituting the CNN shares multiply-accumulate unit (MACU) and post-processing module (PPM) to perform operations. The operation result is transferred from the FPGA accelerator to the DRAM using the AXI_MEM bus.

2) PARALLELIZED PROCESSING MODULE

The general-purpose embedded CPU is very slow to perform convolution operations with six nested loops. To solve this problem, the CNN inference process was accelerated by implementing parallelized processing modules in the PL area of the SoC platform. It was necessary to find a part where the parallelism strategy could be applied in six nested loops to implement a parallelized processing module. Two among several possible parallelism strategies were used to implement the loop-level parallelism efficiently on embedded devices [46].

First, the intra-convolution parallelism strategy was invoked. Intra-convolution parallelism was achieved by fully unrolling the loop corresponding to the MACU internal operation shown in Fig. 5. The fully unrolled loop was implemented with a MACU consisting of nine parallelized multipliers and eight adders for the 3×3 convolution operations. Second, the inter-feature map parallelism strategy was applied. MACUs shown in Fig. 5 were used to achieve inter-feature map parallelism by partially unrolling loops for different channels of the output feature map. Because the unroll factor of the partially unrolled loop was 4, the accelerator could be executed MAC operations in parallel for four other channels of the output feature map. The 1×1 operations are performed by reusing a part of the fully unrolled MACU for 3×3 convolution operations.

In given the SoC platform, there were opportunities to use more parallelism strategies. However, in the case of intra-feature map parallelism, it was not free from data dependency because all the results of the MAC operation for each input channel targeting the pixels of the same output channel had to be accumulated; thus, a great deal of resources were likely to be utilized. In the case of inter-convolution parallelism, it was not suitable for implementation in embedded devices because a large amount of data would have to be loaded in the on-chip buffers in advance.

3) DATA CACHING MODULE

The process of accessing data from off-chip memory every time for MAC operation is expensive both in terms of latency and energy. Moreover, the on-chip memory inside the PL area is not large enough to store all the parameters and calculation results of a CNN. Therefore, there is a need to minimize the number of accesses to off-chip memory by maximizing data reuse in the on-chip memory. Three data caching modules (FMCM, PCM, OCAM) were used in the PL area.

In the MACU with the intra-convolution parallelism strategy applied, the 3×3 convolution filter requires 9 pixels of the input feature map for MAC operation. For efficient prefetching of the input feature map, the accelerator needs line buffers that can store at least 3 rows. Line buffers that could store 4 rows of the input feature map were made for convenience in addressing. These line buffers were bundled

together and called a feature map caching module (FMCM). To maximize parameter data reuse, an on-chip memory was created that can store all parameters of the layer in which the operation is performed, which memory is called the parameter caching module (PCM). In addition, the PCM for each processing module was divided to use the inter-feature map parallelism strategy. A memory was created to accumulate the MAC operation results for the channel of the output feature map during MAC operation, which memory is called output channel accumulation module (OCAM).

4) POST-PROCESSING MODULE

All the remaining operations except convolution operations are performed in the PPM. The PPM consists of the shift operations, add operations, and sign-bit comparison operations. The PPM receives the final output value of the OCAM as an input. Moreover, like the parallelized processing module, the inter-feature map parallelism strategy is applied with the unroll factor set to 4. The final output of the PPM is writeback to the DRAM to enter the input of the next layer.

5) TASK-SCHEDULING

Fig. 6 shows task-scheduling that optimizes the dataflow of each of the parallelized modules constituting the CNN accelerator. First, the CPU calculates the size of the data caching module and the address of the layer and data region using the pre-calculated network constraints before starting the CNN operations. The computed addresses and layer configurations are transferred from the PS to the PL configuration registers through the AXI_LITE bus. Data caching modules (FMCM, PCM) prefetch the data from DRAM through the AXI_MEM bus using the pre-transferred address and layer configurations. The prefetched data is loaded into the pixel buffer and parameter buffer, and the MAC operation is performed in the parallelized processing module. The OCAM stores the intermediate results of the MAC operation and performs accumulation. After the accumulation is completed, the PPM executes the remaining operations. This dataflow lasts until all processes in the layer are terminated.

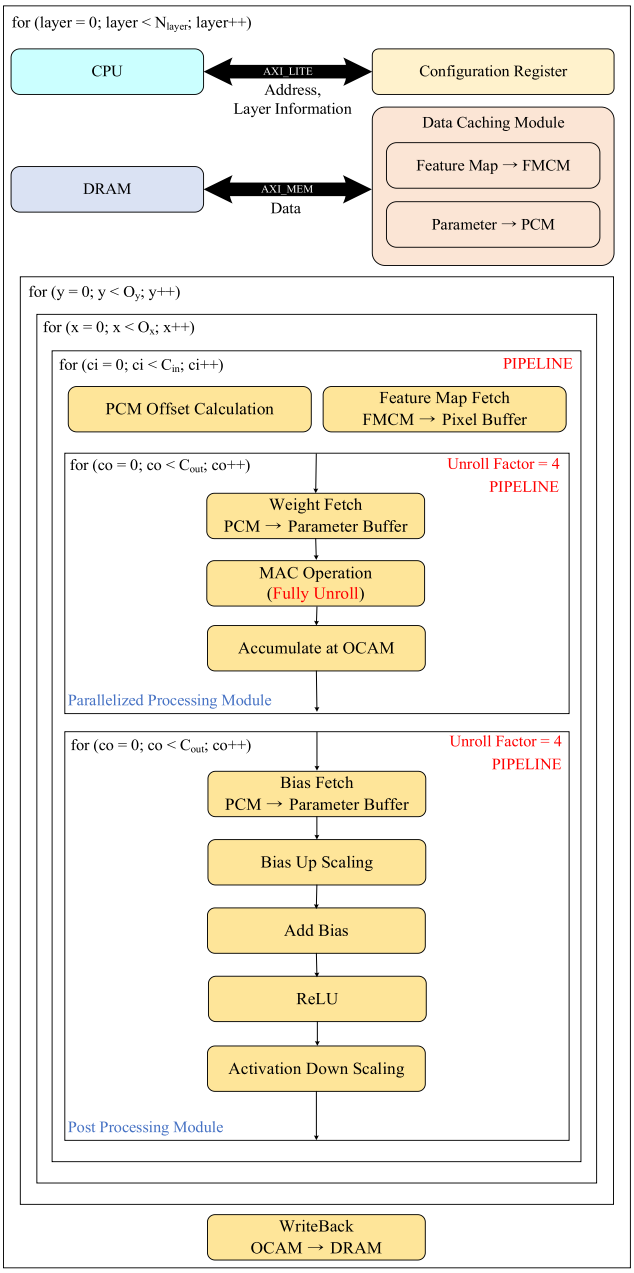

FIGURE 6. Overall task-scheduling for the proposed CNN accelerator.

IV. EXPERIMENTAL RESULTS

In this section, the experimental environment and results are described that were set to evaluate the performance of the proposed CNN optimized for embedded systems and the proposed hardware-friendly quantization method. All evaluations and measurements were performed on a workstation equipped with Intel®Xeon®Gold 512 CPU @ 2.2 GHz, 180 GB RAM, and an NVIDIA Tesla V100 (SXM2) GPU. A new training dataset was created using the OpenCV library. The CUDA library and Pytorch framework were used for floating-point training, quantization-aware training, and integer-arithmetic-only inference. The number of operations was measured using the framework of [47]. The proposed CNN accelerator for real-time facial emotion recognition was implemented using the Xilinx Zynq-7000 SoC ZC706 Evaluation Kit. It consisted of a dual-core ARM Cortex-A9 processor (PS) and XC7Z045 FPGA (PL).

During floating-point training, the batch size was 64, and the number of epochs was 200. The Adam method was chosen as the optimization algorithm with a learning rate of 1e-2 and weight decay of 1e-5. The quantization-aware training methods used in the experiment herein included LSQ [38], TQT [39] and the proposed LLTQ. The hyperparameters of the three quantization methods were set as follows: the batch size was 32 and the number of epochs was 100. The initial learning rate was set to 1e-5 for weights and

FIGURE 7. Some samples for the FERPlus-A dataset: (a) Original image. (b) Bilateral filter, CLAHE applied to (a). (c) Sharpening filter, CLAHE applied to (a). (d) Bilateral filter, CLAHE, sharpening filter applied to (a).

biases, and 1e-3 for scale factors, log-thresholds, and level-thresholds.

A. RESULTS FOR THE NEW TRAINING DATASET

In real-world situations, it cannot be assumed that only images of good quality become CNN input. Therefore, data were added considering various conditions to assure high performance of the CNN in real-world situations. A bilateral filter [11] was applied to add noise-removed and border-preserved face images. CLAHE [12] was used to add high-contrast images and appropriate contrast images. By applying a sharpening filter, images were added that emphasize the boundary features of the face. In this way, FERPlus-A was constructed. This is a dataset that can build a more robust CNN in real-world situations. The FERPlus-A training dataset includes 143,545 images that were added to the 28,709 FERPlus training dataset, resulting in 172,254 images. Fig. 7 shows some samples from the FERPlus-A training dataset. The CNN proposed in Section III-B was used to evaluate whether the FERPlus-A dataset showed improved performance. The CNN was trained using the FERPlus and FERPlus-A training dataset and their performance was evaluated using the same FERPlus test dataset. When trained using the FERPlus training dataset, the facial emotion recognition accuracy was approximately 71.85%, and when trained with the FERPlus-A training dataset, the accuracy was approximately 86.58%. A performance improvement of approximately 14.73% was confirmed when using the FERPlus-A training dataset.

B. COMPARISON OF OTHER QUANTIZATION WORKS

Eight-bit quantization-aware training was performed for all three quantization methods. Table 1 shows the experimental results. Compared to the floating-point, the method of [38] improved accuracy by approximately 0.13%, but because the scale factor was still a floating-point type, the method was not hardware-friendly. This is because it still required a dedicated operator. The method of [39] is hardware-friendly, but it reduced accuracy by approximately 0.45% compared to the floating-point result. After quantization-aware training using LLTQ, CNN inference was conducted using only integer-arithmetic. As a result of applying LLTQ, it was confirmed that there was no accuracy drop relative to the floating-point method. Additional experiments of the LLTQ are provided in Appendix B of the supplementary material document.

C. COMPARISON WITH OTHER LIGHTWEIGHT CNNs

The performance of the proposed lightweight CNN optimized for embedded devices was compared with other facial emotion recognition studies using a lightweight CNN. Comparison of facial emotion recognition accuracy, the number of operations, and parameter size are shown in Table 2. The proposed CNN parameter size is 0.39 MB, the number of operations is 28 M IOPs, and the accuracy is approximately 86.58%. The accuracy of facial emotion recognition by the proposed CNN achieved the highest facial emotion recognition accuracy among the other methods considered [14]–[23]. As shown in Table 2, the new method outperforms the methods of [14]–[16], [18]–[21], [23]-2, [23]-3 in terms of accuracy, the number of operations, and parameter size. In a comparison with the method of [22], it was not possible to measure the number of operations and parameters, but the proposed method exhibits 5.07% higher accuracy. Compared to [17], the new method is 6.33 times smaller parameter size and 3.28% more accurate. Although its number of operations is 1.4 times larger, it is more efficient because it uses only integer-arithmetic. Compared with the method of [23]-1, the proposed method is 1.05 times larger parameter size but 2.52% more accurate. Compared with the method of [23]-2, the new method is 2.15 times smaller and 1% more accurate. Regarding the method of [23]-3, the new approach is 1.67 times smaller, uses 6.07 times fewer operations, and has 0.91% greater accuracy. These results indicate that the proposed CNN achieved the best trade-off between memory usage, number of operations, and accuracy.

D. HARDWARE IMPLEMENTATION ON FPGA

HLS is a way of designing hardware using a high-level language. Implementation details such as state machine and interface are handled by the HLS compiler, so an abstract design can be possible, code can be shared during design and simulation, and rapid deployment is possible. Taking note

TABLE 1. Comparison of three different quantization-aware training methods.

Method	Bit-Width (W/A/B)	Scale factor	Accuracy (%)	vs Baseline	# of Operations	Remark
Baseline	32/32/32	-	86.58	-	28M FLOPs	Floating-point
TQT [39]	8/8/8	Power-of-two	86.13	-0.45	28M FLOPs	Need de-quant, Need pre-calibration
LSQ [38]	8/8/8	Floating-point	86.71	+0.13	28M FLOPs	Need de-quant
LLTQ (This work)*	8/8/8	Power-of-two	86.58	-	28M IOPs	Integer-arithmetic-only

* Inference result using integer-arithmetic after quantization-aware training

TABLE 2. Comparison with other facial emotion recognition studies using a lightweight CNN.

Method	Precision	Input Resolution	Accuracy (%)	Parameter Size (MB)	FLOPs (M)
[21]	FP32	224	86.54	33.23	204
[23]-3	FP32	48	85.67	0.65	170
[23]-2	FP32	48	85.58	0.84	-
[20]	FP32	48	85.29	5.40	394
[14]	FP32	64	84.89	33.42	1,940
[19]	FP32	64	84.80	1.37	70
[18]	FP32	64	84.47	1.37	70
[16]	FP32	32	84.30	3.62	288
[15]	FP32	48	84.30	1.22	190
[23]-1	FP32	48	84.06	0.37	-
[17]	FP32	64	83.30	2.47	20
[22]	FP32	64	81.51	-	-
This work	INT8	64	86.58**	0.39	28*

* Integer Operations (IOPs)
** The best two results are highlighted in red and blue colors, respectively.

TABLE 3. Resource utilization and power consumption of the proposed CNN accelerators.

Resource / Power		Type-1 (FP32)	Type-2 (INT8)	Area / Power Reduction Ratio (Type-1 vs Type-2)
LUT (Avail. 218,600)		16,453	5,090	×3.23 ↓
FF (Avail. 437,200)		33,326	7,588	×4.39 ↓
BRAM (Avail. 1,090)		120	61	×1.97 ↓
DSP (Avail. 900)		205	49	×4.18 ↓
Power (W)	PS	1.670	1.670	-
	PL	2.398	0.415	×5.78 ↓
	Others	0.244	0.225	×1.08 ↓
	Total	4.312	2.310	×1.87 ↓

of these advantages, HLS was chosen for the design. The CNN accelerator IP was designed using Vivado HLS 2019.2. The target clock frequency was 250 MHz. In Vivado 2019.2, the designed IP was connected to the PS and then synthesis and implementation were carried out. Petalinux 2019.2 was used to create the boot image and Linux kernel. A device driver was developed, and real-time facial emotion recognition application was built using Vitis 2019.2. The operation of the CNN accelerator was verified at the terminal connected to the UART.

Here, two types of FPGA-based CNN accelerators are shown in Table 3: Type-1 using 32-bit floating-point parameters (FP32) and Type-2 using 8-bit integer parameters (INT8) that quantized. Table 3 shows the resource utilization and power consumption of the proposed CNN accelerators. When comparing the area reduction ratio of Type-2 to that of Type-1, it was possible to reduce LUTs by 3.23 times, FFs by 4.39 times, BRAMs by 1.97 times, and DSPs by 4.18 times. Furthermore, the power was measured using the Vivado power estimator. The three factors that cause power consumption are the power generated by the CPU in the PS area, the power generated by the CNN accelerator IP in the PL area, and other power. The power consumption of the PS area that occurs in Type-1 and Type-2 is the same. However, the power of the PL area decreased by 5.78 times and the other power decreased by 1.08 times; thus, the total power decreased by 1.87 times. As shown in Table 4, The proposed CNN accelerator was compared to previous implementations. Compared with the method of [40], the new Type-2 method was found to be more accurate, to use fewer resources, and to have frame rate approximately 25 times faster. Compared to the work in [41], the new Type-2 approach is more accurate, uses 2.2 times fewer DSPs, and has a frame rate about 20 times faster. Compared to the accelerator in [42], the new Type-2 CNN accelerator is more accurate, but has a frame rate of about 5 FPS slower. Even so, the new accelerator

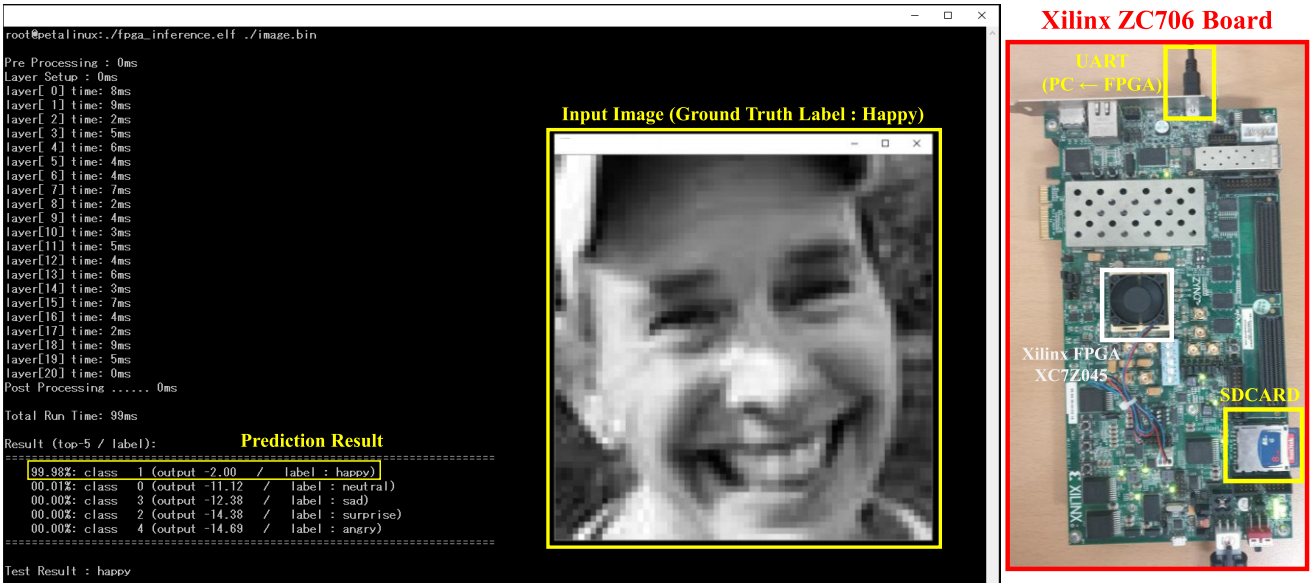

FIGURE 8. Demonstration of an FPGA-based CNN accelerator using only 8-bit integer-arithmetic for real-time facial emotion recognition.

TABLE 4. Comparison with other FPGA-based CNN accelerators for facial emotion recognition.

Method	[40]	[41]	[42]	[43]	[44]	Type-1 (This work)	Type-2 (This work)
Test Dataset	FER2013	FER2013	FER2013	-	-	FERPlus	FERPlus
Accuracy (%)	60.3	65.0	66.0	-	-	86.58	86.58
Design Entry	HLS	RTL	RTL	RTL	RTL	HLS	HLS
FPGA Vendor	Xilinx	Altera	Altera	Altera	Altera	Xilinx	Xilinx
Platform	XC7Z020	DE-10	DE-10	Cyclone-V	Cyclone-V	XC7Z045	XC7Z045
Bit-Width	FP32	FP32	INT16	FX16	FX16	FP32	INT8
Frequency (MHz)	200	200	130	125	50	250	250
LUT or ALM	36K	-	22K	61K	52K	16K	5K
FF	24K	-	-	47K	58K	33K	7K
BRAM	118	-	553	171	63	120	61
DSP	152	108	112	274	270	205	49
Power (W)	-	-	-	-	-	4.31	2.31
Frame Rate (FPS)	0.4	0.5	15*	-	-	7**	10**

* The best two results are highlighted in red and blue colors, respectively.
** FPS was measured by test dataset with the batch size of 1 and averaged them.

uses about 4.41 times fewer LUTs, 9.07 times fewer BRAMs, and 2.29 times fewer DSPs. Compared with the methods in [43] and [44], the proposed method uses fewer resources. The runtime profile table of the proposed CNN accelerator is provided in Appendix C of the supplementary material document. The results show that the proposed accelerator achieved real-time face emotion recognition while consuming low power and utilizing low resources. This has great advantages as a real-time facial emotion recognition application that requires high classification accuracy in embedded systems.

V. CONCLUSION

In this paper, a new training dataset is presented that was created by combining various image processing algorithms to improve the performance of facial emotion recognition. Furthermore, a lightweight CNN architecture that optimizes memory usage and the number of operations for embedded devices was proposed. The use of log level threshold quantization (LLTQ) was also proposed. This is a novel hardware-friendly quantization method that uses only integer-arithmetic by addressing the problems of existing methods. After applying the proposed methods, the proposed

CNN was evaluated using the FERPlus test dataset. The accuracy of the proposed CNN for facial emotion recognition was approximately 86.58% and exhibited no accuracy drop compared to the floating-point approach. The parameter size was 0.39 MB and the number of operations was 28 M IOPs. A CNN accelerator for real-time facial emotion recognition was implemented on the SoC platform and used 5,090 LUTs, 7,588 FFs, 61 BRAMs, and 49 DSPs. It consumes a power of about 2.3 W and demonstrates a frame rate of about 10 FPS. We believe that our proposed CNN accelerator design methodology for real-time applications in embedded systems can be applied to other CNN accelerator designs in the same manner.

REFERENCES

[1] P. Ekman and W. V. Friesen, "Constants across cultures in the face and emotion," *J. Pers. Soc. Psychol.*, vol. 17, no. 2, pp. 124–129, 1971.
[2] P. Ekman and W. V. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto, CA, USA: Consulting Psychologists Press, 1978.
[3] Y. Huang, F. Chen, S. Lv, and X. Wang, "Facial expression recognition: A survey," *Symmetry*, vol. 11, no. 10, p. 1189, Sep. 2019.
[4] S. Zhang, L. Li, and Z. Zhao, "Facial expression recognition based on Gabor wavelets and sparse representation," in *Proc. IEEE 11th Int. Conf. Signal Process.*, Oct. 2012, pp. 816–819.
[5] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Proc. Eur. Conf. Comput. Vis.*, Berlin, Germany, 2004, pp. 469–481.
[6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Comput. Vis. Image Understand.*, vol. 61, no. 1, pp. 38–59, Jan. 1995.
[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2001, pp. 511–518.
[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
[9] M. E. Jabon, J. N. Bailenson, E. Pontikakis, L. Takayama, and C. Nass, "Facial expression analysis for predicting unsafe driving behavior," *IEEE Pervas. Comput.*, vol. 10, no. 4, pp. 84–95, Apr. 2011.
[10] E. Z. B. Zheng, A. Swanson, J. Crittendon, Z. Warren, and N. Sarkar, "Understanding how adolescents with autism respond to facial expressions in virtual reality environments," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 4, pp. 711–720, Apr. 2013.
[11] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, Bombay, India, Jan. 1998, pp. 839–846.
[12] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphics Gems*. New York, NY, USA: Academic, Aug. 1994, pp. 474–485.
[13] 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," 2016, *arXiv:1602.07360*. [Online]. Available: https://arxiv.org/abs/1602.07360
[14] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," in *Proc. 18th ACM Int. Conf. Multimodal Interact.*, Oct. 2016, pp. 279–283.
[15] M. Li, H. Xu, X. Huang, Z. Song, X. Liu, and X. Li, "Facial expression recognition with identity and emotion joint learning," *IEEE Trans. Affect. Comput.*, vol. 12, no. 2, pp. 544–550, Apr. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8528894
[16] A. Wikanningrum, R. F. Rachmadi, and K. Ogata, "Improving lightweight convolutional neural network for facial expression recognition via transfer learning," in *Proc. Int. Conf. Comput. Eng., Netw., Intell. Multimedia (CENIM)*, Nov. 2019, pp. 1–6.
[17] R. K. Pandey, S. Karmakar, A. G. Ramakrishnan, and N. Saha, "Improving facial emotion recognition systems using gradient and Laplacian images," 2019, *arXiv:1902.05411*. [Online]. Available: https://arxiv.org/abs/1902.05411
[18] R. K. Pandey, S. Karmaker, A. G. Ramakrishnan, and N. Saha, "Improving facial emotion recognition systems with crucial feature extractors," in *Proc. 20th Int. Conf. Image Anal. Process.*, Trento, Italy, Sep. 2019, pp. 268–279.
[19] R. K. Pandey, A. G. Ramakrishnan, and S. Karmakar, "Effects of modifying the input features and the loss function on improving emotion classification," in *Proc. IEEE Region 10th Conf.*, Kochi, India, Oct. 2019, pp. 1159–1162.
[20] S. Saurav, P. Gidde, R. Saini, and S. Singh, "Dual integrated convolutional neural network for real-time facial expression recognition in the wild," *Vis. Comput.*, vol. 37, no. 1, pp. 1–14, Feb. 2021.
[21] S. Miao, H. Xu, Z. Han, and Y. Zhu, "Recognizing facial expressions using a shallow convolutional neural network," *IEEE Access*, vol. 7, pp. 78000–78011, 2019.
[22] Z. Lian, Y. Li, J. Tao, J. Huang, and M. Niu, "Region based robust facial expression analysis," in *Proc. 1st Asian Conf. Affect. Comput. Intell. Interact. (ACII Asia)*, May 2018, pp. 1–5.
[23] G. Zhao, H. Yang, and M. Yu, "Expression recognition method based on a lightweight convolutional neural network," *IEEE Access*, vol. 8, pp. 38528–38537, 2020.
[24] I. J. Goodfellow, D. Erhan, and P. L. Carrier, "Challenges in representation learning: A report on three machine learning contests," in *Proc. Int. Conf. Neural Inf. Process.*, Berlin, Germany, Nov. 2013, pp. 117–124.
[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–14.
[26] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," 2014, *arXiv:1406.4773*. [Online]. Available: https://arxiv.org/abs/1406.4773
[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[29] Y. Wen, K. Zhang, K. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 499–515.
[30] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2261–2269.
[31] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 1325–1334.
[32] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Seoul, South Korea, Oct. 2019, pp. 1325–1334.
[33] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," 2018, *arXiv:1810.05723*. [Online]. Available: https://arxiv.org/abs/1810.05723
[34] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "ZeroQ: A novel zero shot quantization framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13169–13178.
[35] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*. [Online]. Available: https://arxiv.org/abs/1805.06085
[36] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[37] S. Jung, C. Son, S. Lee, J. Son, J. J. Han, Y. Kwak, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 4350–4359.
[38] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," 2019, *arXiv:1902.08153*. [Online]. Available: https://arxiv.org/abs/1902.08153

[39] S. R. Jain, A. Gural, M. Wu, and C. H. Dick, "Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks," in *Proc. Mach. Learn. Syst.*, Mar. 2020, pp. 112–128.
[40] H. Phan-Xuan, T. Le-Tien, and S. Nguyen-Tan, "FPGA platform applied for facial expression recognition system using convolutional neural networks," *Procedia Comput. Sci.*, vol. 151, pp. 651–658, Jan. 2019.
[41] T. N. D. Phuc, N. N. Nhut, N. Trinh, and L. Tran, "Facial expression recognition system using FPGA-based convolutional neural network," in *Research in Intelligent and Computing in Engineering*, vol. 1254. Singapore: Springer, Jan. 2021, pp. 341–351.
[42] P. T. Vinh and T. Q. Vinh, "Facial expression recognition system on SoC FPGA," in *Proc. Int. Symp. Electr. Electron. Eng. (ISEE)*, Ho Chi Minh City, Vietnam, Oct. 2019, pp. 1–4.
[43] R. Ding, G. Su, G. Bai, W. Xu, N. Su, and X. Wu, "A FPGA-based accelerator of convolutional neural network for face feature extraction," in *Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC)*, Xi'an, China, Jun. 2019, pp. 1–3.
[44] R. Ding, X. Tian, G. Bai, G. Su, and X. Wu, "Hardware implementation of convolutional neural network for face feature extraction," in *Proc. IEEE 13th Int. Conf. ASIC (ASICON)*, Chongqing, China, Oct. 2019, pp. 1–4.
[45] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*. [Online]. Available: https://arxiv.org/abs/1308.3432
[46] K. Abdelouahab, M. Pelcat, J. Serot, and F. Berry, "Accelerating CNN inference on FPGAs: A survey," 2018, *arXiv:1806.01683*. [Online]. Available: https://arxiv.org/abs/1806.01683
[47] L. Zhu. *THOP: PyTorch-OpCounter*. Accessed: Apr. 26, 2021. [Online]. Available: https://github.com/Lyken17/pytorch-OpCounter

JAEMYUNG KIM received the B.S. degree in electronics engineering from Inha University. He is currently pursuing the M.S. degree in electrical and computer engineering. His research interests include system-on-chip design, field-programmable gate array-based design, and convolutional neural network optimization and acceleration.

JIN-KU KANG (Senior Member, IEEE) received the B.S. degree from Seoul National University, Seoul, South Korea, in 1983, the M.S. degree in electrical engineering from New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and the Ph.D. degree in electrical and computer engineering from North Carolina State University, Raleigh, NC, USA, in 1996. From 1983 to 1988, he was with Samsung Electronics, Inc., South Korea, where he was involved in memory design. In 1988, he was with Texas Instruments, South Korea. From 1996 to 1997, he was with Intel Corporation, Portland, OR, USA, as a Senior Design Engineer, where he was involved in high-speed I/O and timing circuits for processors. Since 1997, he has been with Inha University, Incheon, South Korea, where he is currently a Professor and leads the System IC Design Laboratory, Department of Electronics Engineering. His research interest includes high-speed/low-power mixed-mode circuit design for high-speed serial interfaces.

YONGWOO KIM (Member, IEEE) received the B.S. and M.S. degrees from Inha University, Incheon, South Korea, in 2007 and 2009, respectively, and the Ph.D. degree in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2019. From 2009 to 2017, he was a Senior Engineer with Silicon Works Company Ltd., Daejeon. From 2019 to 2020, he was a Senior Researcher with the Artificial Intelligence Research Division, Korea Aerospace Research Institute, Daejeon. Since 2020, he has been with Sangmyung University, Cheonan, South Korea, where he is currently an Assistant Professor with the Department of System Semiconductor Engineering. His current research interests include image/video processing algorithm, super-resolution, and deep learning hardware architecture for vision processing.