sensors

MDPI

Article

FPGA Implementation for Odor Identification with Depthwise Separable Convolutional Neural Network

Zhuofeng Mo, Dehan Luo, Tengteng Wen *, Yu Cheng and Xin Li

School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China; mozf@mail2.gdut.edu.cn (Z.M.); dehanluo@gdut.edu.cn (D.L.); chengyu@gdut.edu.cn (Y.C.); lixin@mail2.gdut.edu.cn (X.L.)

* Correspondence: wentt@gdut.edu.cn

Abstract: The integrated electronic nose (e-nose) design, which integrates sensor arrays and recognition algorithms, has been widely used in different fields. However, the current integrated e-nose system usually suffers from the problem of low accuracy with simple algorithm structure and slow speed with complex algorithm structure. In this article, we propose a method for implementing a deep neural network for odor identification in a small-scale Field-Programmable Gate Array (FPGA). First, a lightweight odor identification with depthwise separable convolutional neural network (OI-DSCNN) is proposed to reduce parameters and accelerate hardware implementation performance. Next, the OI-DSCNN is implemented in a Zynq-7020 SoC chip based on the quantization method, namely, the saturation-flooring KL divergence scheme (SF-KL). The OI-DSCNN was conducted on the Chinese herbal medicine dataset, and simulation experiments and hardware implementation validate its effectiveness. These findings shed light on quick and accurate odor identification in the FPGA.

Keywords: electronic nose; odor identification; depthwise separable convolutional neural network; FPGA-implementation

Citation: Mo, Z.; Wen, T.; Luo, W.; Cheng, Y.; Li, X. FPGA Implementation for Odor Identification with Depthwise Separable Convolutional Neural Network. *Sensors* **2021**, *21*, 832. https://doi.org/10.3390/s21030832

Academic Editor: Chiman Kwan
Received: 18 December 2020
Accepted: 24 January 2021
Published: 27 January 2021

1. Introduction

An electronic nose (e-nose), which mimics biological olfaction, is an odor analysis device composed of a carefully selected sensor array and an appropriate pattern recognition algorithm [1]. E-nose has the ability to detect and distinguishing characteristics such as the type and concentration of gas. E-nose has become more and more widely used in the world. Sigfredo et al. designed an array of gas sensor and five machine learning algorithms to detect and evaluate contamination in grapevine berries and taint in wines [2]. Hao Wei et al. used PEN3 e-nose for data collection and designed a back-propagation neural network (BPNN) to detect brown core in the Chinese pear variety huangguan [3]. Winston Li et al. used four classifiers—MLP, SVM, KNN, and Parzen—and fusion in Dempster–Shafer to improve the accuracy of odor classification [4]. In addition, fluctuation enhanced sensing (FES) has been applied to the field of e-noses to detect gas-phase chemicals [5–7]. In FES, noise is considered to carry much useful information, so it uses microfluctuations caused by the interaction between chemical sensors and gas molecules to improve sensitivity and selectivity [8]. With the development of e-nose technology, e-nose has been designed and optimized many times and widely used in food testing [9], environmental monitoring [10,11], medical diagnosis [12–14], and the space shuttle [15–17]. In particular, for a long time, Chinese herbal medicine has been classified based on traditional identifying methods such as human smell, taste, vision, and touch, while the human smell is the most common method to distinguish the variety and various herb-growing areas. However, human identification of Chinese herbal medicines is highly subjective and time-consuming because individual differences and external disturbances may easily influence humans distinguishing. In recent years, several new technologies have emerged to replace

manual classification of Chinese medicinal materials, such as gas chromatography-mass spectrometry (GC–MS) [18] and Electronic tongue (E-tongue) [19]. They are challenging to be commonly used because of their high price or damage to the integrity of Chinese medicinal materials.

At present, many deep learning methods and applications have been proposed, and they were also applied to odor identifications. Danli Wu et al. used a convolutional neural network (CNN) to predict odor pleasantness [20]. You Wang et al. designed an optimized deep convolutional neural network to classify dendrobium data collected by e-nose [21]. Yan Shi et al. designed a CNN-SVM model to classify beer data collected by PEN3 e-nose [22]. Deep learning algorithm can automatically extract and recognize odor features from odor data. Compared with traditional machine learning methods [23,24], the deep learning method has more advantages when applied in the field of odor recognition and achieved better performance.

An e-nose is usually operating in a separate way where the gas sensing hardware collects response data then transmits them to a computer for identification. Considering the feature of real-time data processing, some studies integrated odor identification function into automatic odor sampling hardware. Zhiyuan Wu et al. designed a low-cost e-nose system and identified cigarettes using random forest [10]. A. Ali et al. proposed a hardware/software co-design approach using the Zynq platform based on principal component analysis [25]. The integration of the e-nose system has several advantages. First of all, the integration makes the system less complicated and increase mobility because the identification algorithm runs locally and avoid using an additional computer. Furthermore, integration reduces data transmission, which makes it easier for real-time odor identification. It can be seen that the integrated e-nose design has become an important research direction. Although the integrated e-nose has many benefits, such as portability, low cost, and miniaturization, there are still some challenges in designing the integrated e-nose. The design of integrated e-nose is limited by chip performance, which leads to simple algorithms can be conducted. A simple algorithm may have a certain effect in some specific applications, but the effect may worsen in a complex odor recognition field. Complex algorithms are limited by the computational performance of simple chips, resulting in slow recognition speed, which leads to many excellent deep learning models that usually need to run on computers with GPUs. Therefore, it is vital to contrive this system capable of low-cost, fast identification, and maintaining the accuracy of identification, especially in the practical application of e-nose.

The main contributions of this paper are as follows. (1) Deep separable convolution is applied in odor identification. We propose a lightweight deep learning odor identification model named OI-DSCNN, which balances the speed and accuracy of the odor identification algorithm. (2) Accelerated deep learning algorithm based on Field-Programmable Gate Array (FPGA) is introduced into odor identification, in which the overall architecture and modules of OI-DSCNN are designed and optimized. Therefore, odor identification is accelerated in FPGA. (3) The SF-KL quantization scheme is designed to reduce FPGA resource consumption and maintain the accuracy.

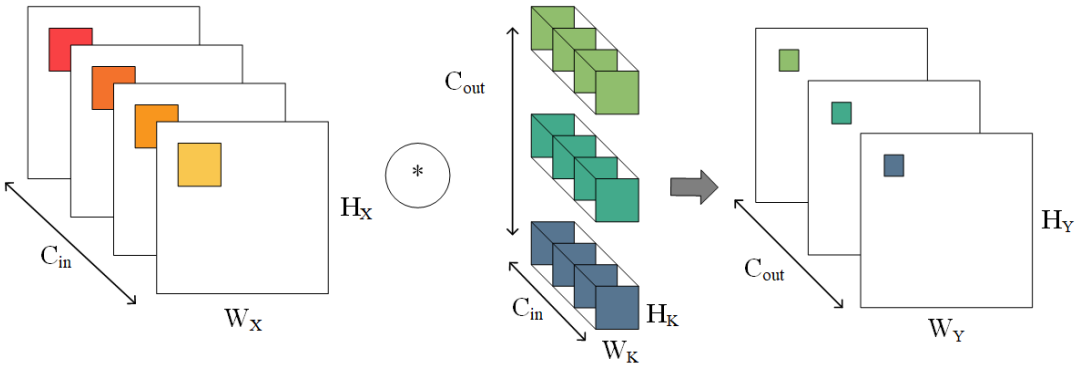
This article is composed as follows. In Section 2, the OI-DSCNN model is first introduced, and then the design and implementation of SF-KL are introduced in detail. In Section 3, the architecture of OI-DSCNN in FPGA, and the design and optimization of each module are illustrated. In Section 4, experiments for the evaluation of the model are demonstrated. In Section 5, conclusions are drawn and future research is prospected.

2. The OI-DSCNN Model

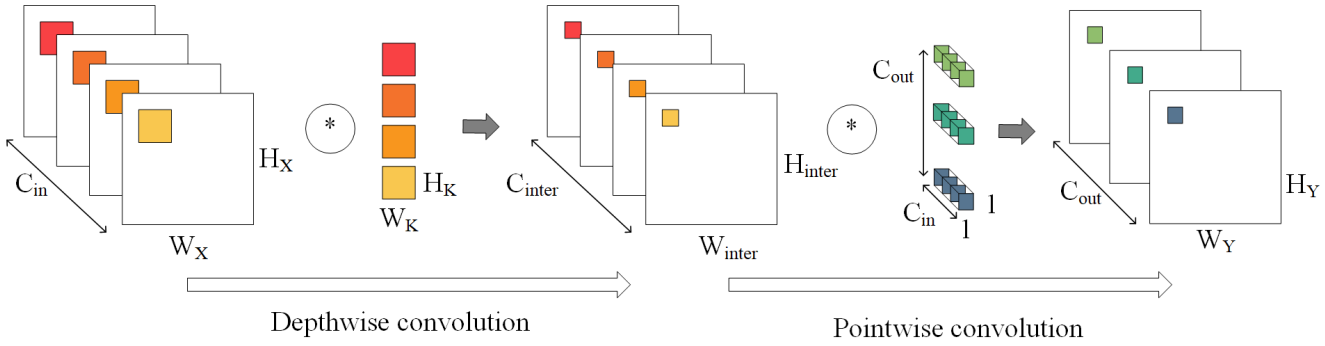
2.1. Depthwise Separable Convolution

The depthwise separable convolution, initially introduced by L. Sifre et al. [26], requires fewer parameters to reduce the high computational burden of standard convolution. It had been applied to frameworks such as Mobilenet [27–29], and it remained sufficiently decent results and performances.

A depthwise separable convolution decomposes a standard convolution into a depthwise convolution and a pointwise convolution. Figure 1 shows the working principle of a standard convolution and depthwise separable convolution.



(**a**) Standard convolution



(**b**) Depthwise separable convolution

Figure 1. The working principle of standard convolution and depthwise seperable convolution. In depthwise convolution, each convolution kernels is convoluted with each input channels. In pointwise convolution, convolution kernel is a 1×1 standard convolution.

As for standard convolutions, as shown in Figure 1a, each input channel requires a convolution that the number of convolution kernels is the same as the output channel. The result of each output channels is the sum of its corresponding convolution kernels and convolutional results of all input channels. Suppose that the size of the input X is $W_X \cdot H_X \cdot C_{in}$, where W_X, H_X, and C_{in} is the width, height, and the number of input channels, respectively. The output Y is $W_Y \cdot H_Y \cdot C_{out}$, where W_Y, H_Y, and C_{out} are the width, height, and the number of output channels, respectively. Compared with the standard convolution, depthwise separable convolution separates the network into depthwise convolution and pointwise convolution, which reduces the number of weights and the amount of computation:

$$\frac{N_{K_{DC}} + N_{K_{PC}}}{N_{K_{SC}}} = \frac{W_K \cdot H_K \cdot C_{inter} + C_{inter} \cdot C_{out}}{W_K \cdot H_K \cdot C_{in} \cdot C_{out}} = \frac{1}{C_{in}} + \frac{1}{W_K \cdot H_K} \tag{1}$$

where $N_{K_{SC}}$ is the size of the standard convolution kernels. $N_{K_{DC}}$ is the size of the depthwise convolution kernels. $N_{K_{PC}}$ is the size of the pointwise convolution kernels. C_{inter} is intermediate channels of depthwise separable convolutional network.

2.2. OI-DSCNN Modeling

The 2-dimensional sensing responses collected from e-noses were used directly in many studies [20,30–35]. In our studies, the $w \times h$ input data were reshaped to $w \times 1 \times h$ for depthwise convolution computations of w independent channels as shown in Figure 2.

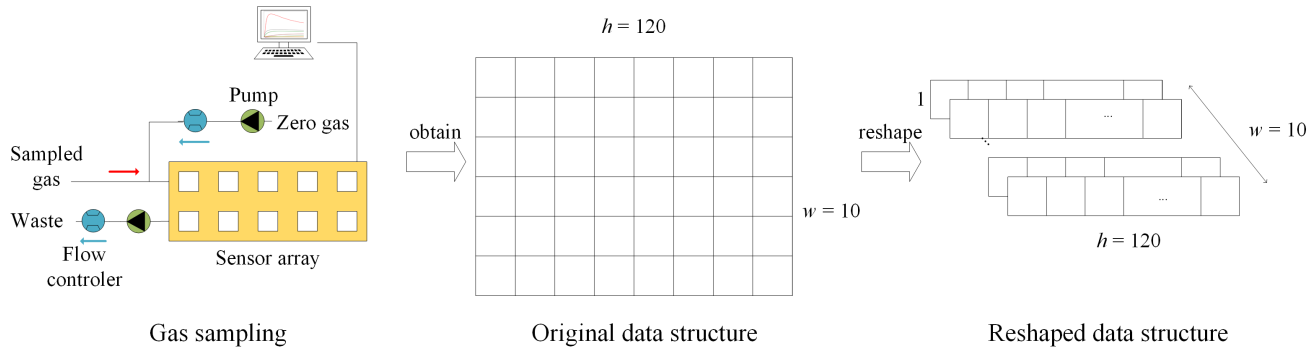


Figure 2. Input data from PEN3 e-nose.

The structure of the OI-DSCNN model is shown in Figure 3 and Table 1. It consists of two depthwise separable convolution layers and a fully connected layer, each depthwise separable convolution layer with a max-pooling layer. Softmax classifier is used for the identification. The two depthwise convolution layers consisted of 1×3 and 1×2 convolution kernels, respectively. Nonlinear activation function *Relu* is selected in all convolutional layers. The stride size of all convolutional layers is set to 1, and the max-pooling layer is set 2.

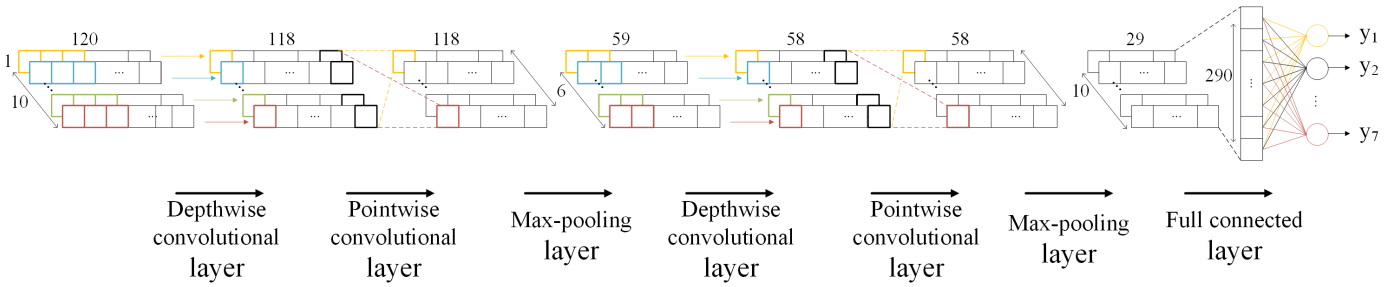


Figure 3. The OI-DSCNN model.

Table 1. the OI-DSCNN model architecture.

Layer	Type	Filter Shape	Input Size
Depthwise separable convolution 1	Depthwise convolution	$1 * 3 * 10$	$1 * 120 * 10$
	Pointwise convolution	$1 * 1 * 10 * 6$	$1 * 118 * 10$
	Max-pooling	$1 * 2$	$1 * 118 * 6$
Depthwise separable convolution 2	Depthwise convolution	$1 * 2 * 6$	$1 * 59 * 6$
	Pointwise convolution	$1 * 1 * 6 * 10$	$1 * 58 * 6$
	Max-pooling	$1 * 2$	$1 * 58 * 10$
Fully connected 3	Fully Connected	$290 * 7$	$1 * 29 * 10$
Classifier	Softmax	-	7

2.3. Quantization

Generally speaking, a 32-bit floating-point arithmetic consumes four times more DSPs than a 8-bit fixed-point arithmetic in parallel operation, and it requires over two times computation timing. Besides, it consumes more block memory (BRAM) and registers resources to store a large amount of parameters and intermediate computing results. To decrease the number of resources used, currently, many effective quantization methods

have been proposed [36–38], which these methods were mostly designed for GPUs and CPUs but not for FPGA. Some common methods can decrease the model accuracy under the same network scale in the quantization phase in FPGA [39,40]. Here, we proposed a saturation-flooring KL_divergence scheme (SF-KL) to reduce the usage of computing devices because of the limited DSP resource in the FPGA chip.

The OI-DSCNN is first trained and validated in floating-point operation, then the proposed SF-KL is used to find the optimized saturation-flooring bits of outputs on each layers. After that, the quantization methods were designed in hardware logic. There are three parts of quantization process as follows.

2.3.1. Quantization of Model Weights

The OI-DSCNN model is fully trained using floating-point algorithms. After the training, parameters were first converted from 32-bit floating-point to 8-bit fixed-point using linear quantization. The following is the procedure of how to quantize weights.

We need to find the maximum weight in each layer:

$$M_i = \max(\text{abs}(P_i)), \tag{2}$$

where $i = 1, ..., m$, m is the number of layers in the model. P_i represents weights in each layer:

$$P_i = \begin{bmatrix} p_1 & p_2 & ... & p_n \end{bmatrix} \tag{3}$$

where n is the number of weights in a layer.

The scale factor sf is obtained by

$$sf_i = \frac{2^8 - 1}{M_i} \tag{4}$$

The quantized value Q_{P_i} of P_i is:

$$Q_{P_i} = \text{round}(P_i \cdot sf_i) \tag{5}$$

2.3.2. Quantization of Input

As the different sensitivities and selectivities to various gases on gas sensors, the input was mean normalized to $[-1, 1]$:

$$\mu_i = \frac{\sum_j x_{i,j}}{120} \tag{6}$$

$$x'_{i,j} = \frac{x_{i,j} - \mu_i}{\max(x_i) - \min(x_i)} \tag{7}$$

where $j = 1, 2, ..., 120, i = 1, 2, ..., 10$. μ_i, $\max(x_i)$ and $\min(x_i)$ are the average value, maximum value, and minimum value of sampling points, respectively. $x_{i,j}$ and $x'_{i,j}$ represent the original input and normalized input, respectively. After that, the input is linearly quantized to the N-bit fixed-point number $Q_{x_{i,j}}$:

$$Q_{x_{i,j}} = \text{floor}\left(x'_{i,j} \cdot \left(2^8 - 1\right)\right) \tag{8}$$

2.3.3. Quantization of Output on Each Layer

In order to reduce the workload of manual attempts and avoid introducing additional parameters, an output quantization is shown in Equation (9). It calculates by the floor and saturation operating, which the scale factor is constrained to $\frac{1}{2^N}$:

$$Q_{Y_N} = \begin{cases} 127 & \frac{Y}{2^N} \geq 127 \\ \text{floor}(\frac{Y}{2^N}) & -128 < \frac{Y}{2^N} < 127 \\ -128 & \frac{Y}{2^N} \leq -128 \end{cases} \quad (9)$$

where Y is the output, and Q_{Y_N} is the quantized output. In this way, the quantization of the output can be operated only by floor and saturation without multipliers, thereby reducing the consumption of DSPs. Algorithm 1 demonstrated the completed procedure of SF-KL obtaining the scale factor.

Algorithm 1 SF-KL scheme

Input: X_n;Q_{X_n};P;Q_P

Output: sf_i

```
/*define Model A and B, while A uses floating-point and B uses integer*/
A, B = OI − DSCNN(Weight, Data)
function QUANTIZATION(X_n, Q_{X_n}, P, Q_P)
    /*j = DWC1, PWC1, ..., FC*/
    for j in OI − DSCNN do
        for k = 1 → n do
            if j ≠ Maxpooling then
                /* X'_k are the input for next layer*/
                X'_k = A.j(P_j, X_k)
                Y_k = B.j(Q_{P_j}, Q_{X_n})
                /*max = the highest bit of the output value - 8*/
                for N = 0 → max do
                    /*KL_divergence*/
                    KL_{N_k}(X'_k, Q_{Y_{N_k}})
                end for
            end if
        end for
        KL_N = ∑_k KL_{N_k}
        /*Find the minimum KL_divergence and get N*/
        N ← min(KL_N)
        /*Get the scale factor*/
        sf_j ← 1/2^N
    end for
end function
```

First, it needs to prepare the original weight of P and the quantized weight Q_P. n original training samples X_n were randomly selected to calculate scale factor $\frac{1}{2^N}$. The selected samples X_n were quantized to Q_{X_n} according to Section 2.3.2. Three samples of each type

are randomly selected, which a total of $n = 3 \times 7 = 21$ samples as input to the algorithm. Two identical OI-DSCNN models are constructed. The difference between the two models is that one uses floating-point calculations and the other uses integer calculations. Then, import the samples into the two models separately. The results calculated by each layer in the two models (except for maximum pooling) will be used to calculate *KL_divergence*:

$$KL\_divergence(X'_k, Q_{Y_N}) = \text{sum}\left(X'_k \cdot \log\left(\frac{X'_k}{Q_{Y_N}}\right)\right) \tag{10}$$

X'_k is the output of floating point, and Q_{Y_N} is the output of integer number after saturation-flooring. Then, the scale factor of the smallest value of *KL_divergence* is used as the input of the next layer of integer number. We use the minimum value of the mean of the *KL_divergence* of different scale factors due to the scale factor. As a nonlinear Relu activation function is used, for the n-bit output $Y[n-1:0]$ in FPGA, the Equation (9) is transformed as

$$Q_{Y_N} = \begin{cases} 8'b0 & \text{if} \quad Y[n-1] = 1 \\ \{0, Y[N+6:N]\} & \text{elseif } (|Y[n-1:N+7]) = 0 \\ \{0, 7'b111\_1111\} & \text{other} \end{cases} \tag{11}$$

In this way, it quantizes the output of each layer to 8 bits so that it can reduce the input bitwidth of the following layers.

3. FPGA Design and Implementation

Figure 4 describes the overview of the system architecture. This system was implemented on the Digilent Arty Z7-7020 development board, USA, which is detailed in Section 4. In order to take advantage of parallel computation in FPGA, the four convolutional layers and the fully connected layer are operated in a pipeline. In order to reduce the data access time and improve the performance of the system, the output of each layer of the network is stored in on-chip memory. The system is composed of PS block, PL block, and DDR memory. The PL block composes of winograd depthwise convolution unit (WDCU), pointwise convolution unit (PCU), fully connected unit (FCU), on-chip memory, and direct memory access (DMA). WDCU, PCU, and FCU are responsible for calculating each convolutional layer, activation layer, max-pooling layer, and fully connected layer. On-chip memory includes input data buffer, intermediate data buffer, and weight buffer. The input data buffer, intermediate data buffer, and the weight buffer for the fully connected layer were designed by block RAM (BRAM), while the weights of convolutional layers are stored in distributed RAM (DRAM). Thank to some techniques used when modeling OI-DSCNN, the number of parameters and computational complexity saw a significant decrease, which the experimental results will be illustrated in detail in Section 4. Besides, the BRAM with 630 KB in a Zynq-7020 chip is sufficient to store such a small scale of several input samples when the model is implemented in the pipeline. Moreover, it is also sufficient to store intermediate results, input data, and parameters in the on-chip memory, which reduces the memory access timing and enhances the performance of system.

The core of the PL design is the convolution layer, which significantly impacts the computation rate of the entire system. Due to the grouped convolution structure of the depthwise convolution layer, the data within each input channel cannot be reused. The Winograd algorithm for depthwise convolution was introduced [41], and the structure of the WDCU was presented in Section S1. WDCU consists of line buffer, input transformation unit, multiplier array unit, configurable output inverse transformation unit, and relu and quantization unit. A line buffer is used to buffer the data transferred to the input transformation unit. The transformed input data are transmitted to the multiplier array unit that multiplies the inputs and the weights from the WDCU weight buffer unit. After the inverse transformation of the Winograd algorithm is completed by the configurable

output inverse transformation unit, the 8-bit output of the depthwise convolutional layer is obtained through relu and quantization unit and transmitted to the WDCU output buffer.

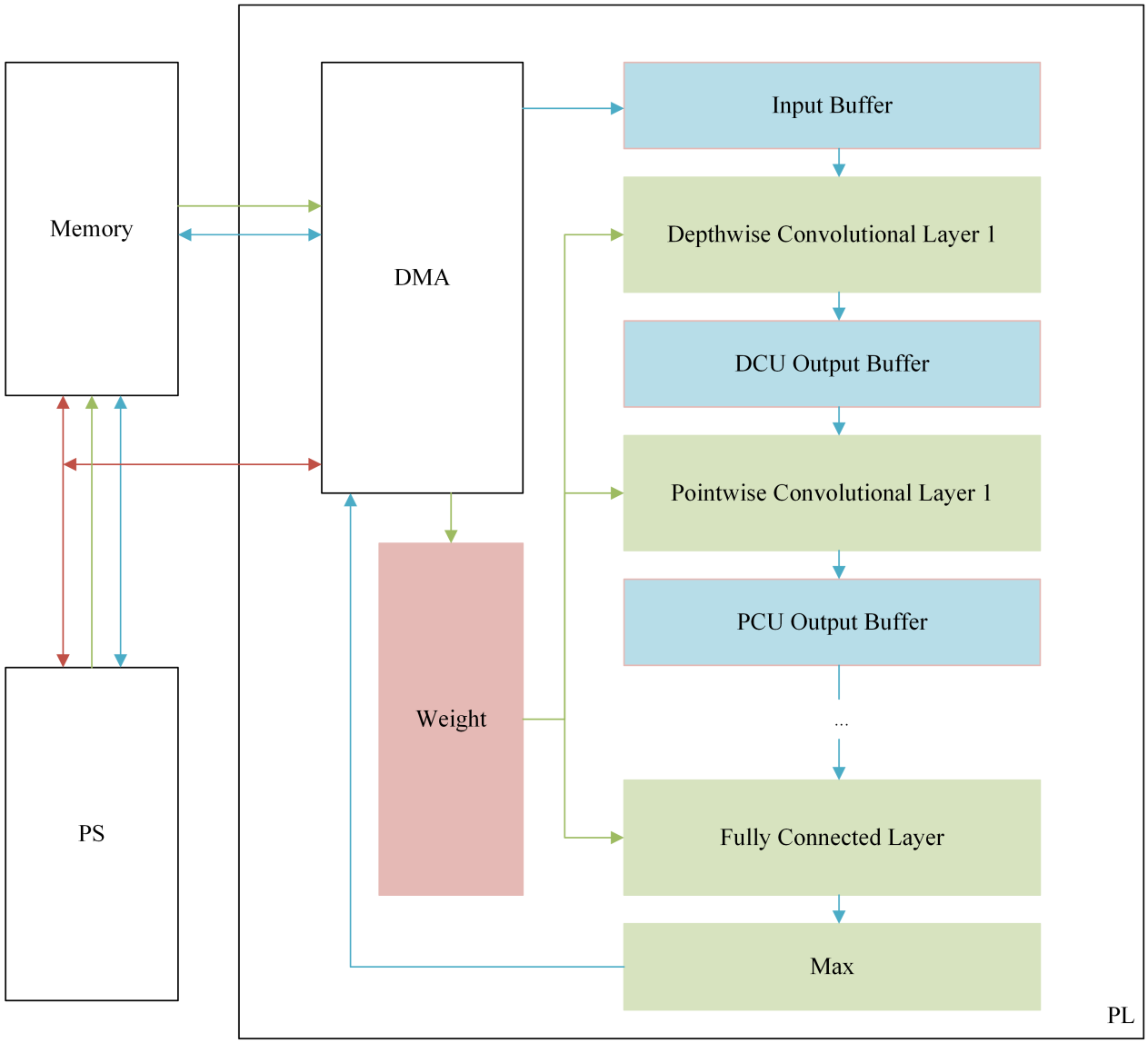


Figure 4. An overview of the FPGA implementation architecture of OI-DSCNN. The red line is the control path. The green line is the weight path. The blue line is the data path.

The PCU, which is used for pointwise convolution, comprises line buffer, multiplier array unit, adder tree unit, relu and quantization unit, and maxpooling unit. The output of WDCU is connected to the line buffer of the first PCU, then passed to the line buffer of the next PCU. Details about the design of PCU can be seen in Section S2. For the fully connected layer, as the weights and data are not reused, a compromise solution is adopted in our design [42,43]. The FCU, which is used to calculate the output of fully connected layer, comprises line buffer, multiplier array unit, adder tree unit, and accumulator unit. FCU divides the data from the PCU output buffer into several 1×5 small scale vectors and obtains the output through multiplier array unit, adder tree unit, accumulator unit, and quantization unit. Details about the design of FCU can be seen in Section S3. The final outputs of FCU are compared in the max unit, and then the comparison result is sent to the PS block.

4. Experiments and Results

4.1. Experimental Setup

The data set was collected from an e-nose, PEN3, AIRSENSE analytics Inc, Germany. PEN3 e-nose is a general gas response signal sampling instrument, which has 10 metal oxide gas sensors, and each sensor has a different sensitivity to different gases, as shown in Table 2. Thanks to a sensor array composed of ten gas sensors, PEN-3 has the ability to sense various gases. The settings of the e-nose are described in Table 3.

Table 2. Sensor array details in PEN-3 e-nose.

Sensor	Sensor Sensitivity and General Description
W1C	Aromatic compounds.
W5S	Very sensitive, broad range of sensitivity, reacts to nitrogen oxides, very sensitive with negative signals.
W3C	Ammonia, used as sensor for aromatic compounds.
W6S	Mainly hydrogen.
W5C	Alkanes, aromatic compounds, less polar compounds.
W1S	Sensitive to methane. Broad range.
W1W	Reacts to sulfur compounds, H2S. Otherwise sensitive to many terpenes and sulfur-containing organic compounds.
W2S	Detects alcohol, partially aromatic compounds, broad range.
W2W	Aromatic compounds, sulfur organic compounds.
W3S	Reacts to high concentrations (>100 mg/kg) of methane–aliphatic compounds.

PEN3 was used to collect gas sensing responses of seven Chinese medicinal materials as dataset: betel, galangal, fructus amomi, fructus aurantii, curcuma zedoary, rhizoma zingiberis, and moutan bark. The experimental environment temperature of the e-nose was 25 ± 0.5 °C, and the humidity was $75 \pm 2\%$. The procedures for preparing these materials were set as follows.

1. The materials were placed in a clean beaker and kept still for more than 20 min.
2. Before collecting the data, the sensor chamber is cleaned and calibrated.
3. Data collection was conducted, and each sample was collected for 120 s.
4. Steps 1–3 were repeated, and 100 samples of each kind of Chinese herbal medicine were collected. The final dataset consisted of 700 data samples in 7 categories, 100 samples for each category, as shown in the Figure 5.

Table 3. Settings of PEN-3 e-nose.

Options	Settings
Sample interval	1.0 s
Presampling time	5.0 s
Zero point trim time	5.0 s
Measurement time	120 s
Flushing time	120 s
Chamber flow	150 mL/min
Initial injection flow	150 mL/min

We compared the OI-DSCNN model with several odor identification model methods. We mainly used PyTorch and scikit-learn in Python to model these algorithms. Pytorch is a deep learning framework for Python [44]. Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms [45].

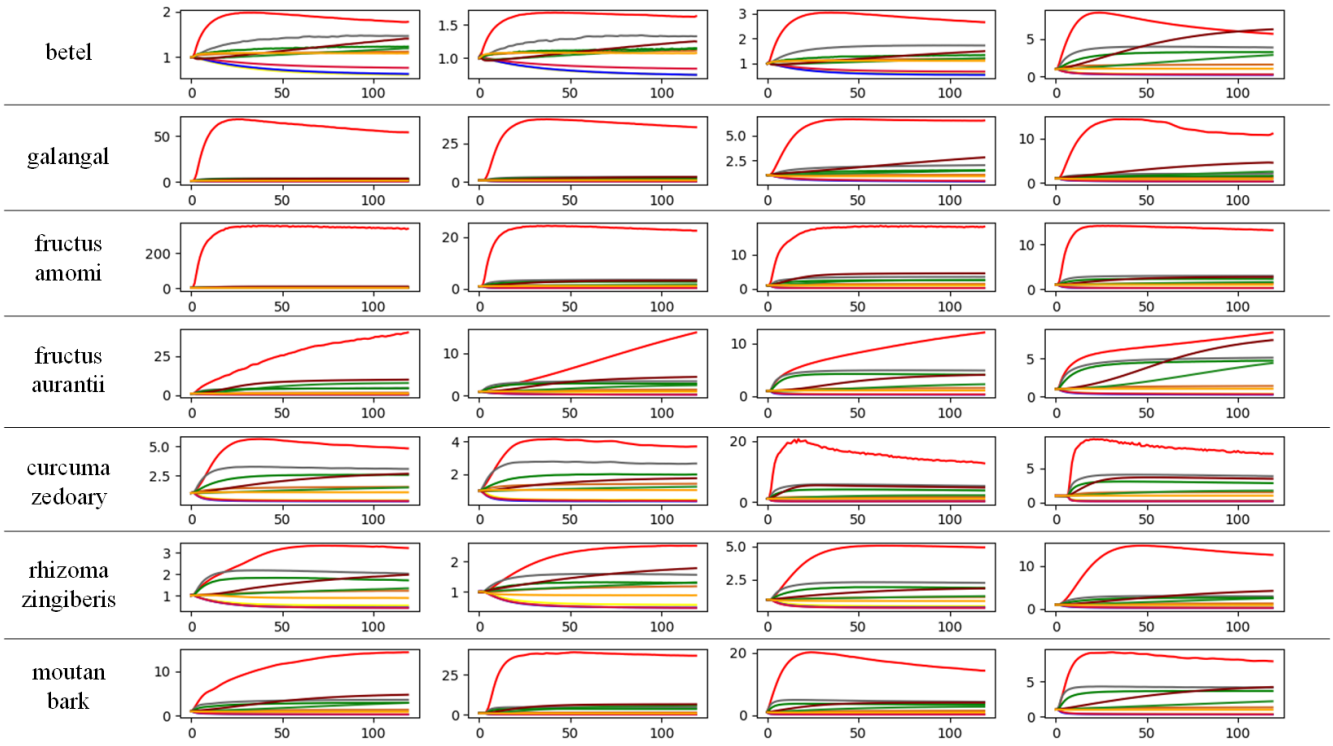


Figure 5. Part of Chinese herbal medicines dataset.

Stochastic Gradient Descent (SGD) was used to train the model. The number of samples per training was 21, and the momentum was 0.9. The learning rate was initially set to 0.01 and adjusted by $ReduceLROnPlateau()$. When the loss of validation set has stopped improving, it was divided by 10 and finally stopped at 0.0001. The loss function was set to $CrossEntropyLoss()$.

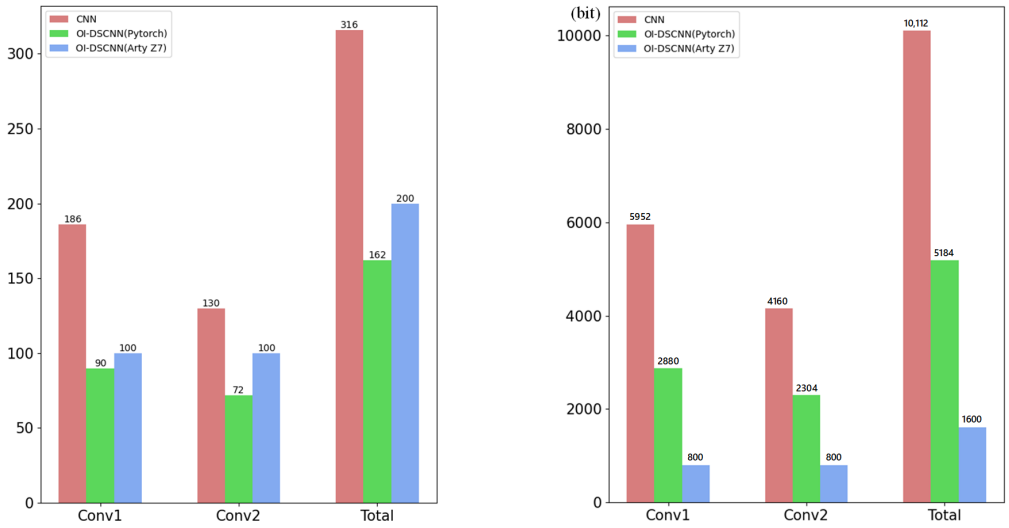
The design was coded in Verilog HDL and synthesized and implemented on Xilinx Vivado. The OI-DSCNN model was evaluated on a Digilent Arty Z7-7020 development board. The Arty Z7-7020 development board contained a Zynq-7020 FPGA chip, a Zynq-7000TM All Programmable System-on-chip (AP SoC). The AP SoC consists of the Programmable Logic (PL) block and the Processing System (PS) block. The Arty Z7-7020 platform characteristics are shown in Table 4. Three different types of platforms, that is, Raspberry Pi 4B, CPU, and GPU with similar or slightly better performances, were selected to compare the performance with FPGA.

Table 4. Some specifications of Arty Z7-7020 development board.

Item	Specification
Product variant	Arty Z7-7020
Zynq part	XC7Z020-1CLG400C
Look-up Tables (LUTs)	53,200
Flip-Flops	106,400
Block RAM	630 KB
DSPs	220
Clock Management Tiles	4
Processor	650 MHz dual-core Cortex-A9 processor 512 MB DDR3 with 16-bit bus @ 1050 Mbps
Memory	16 MB Quad-SPI Flash with factory programmed 48-bit globally unique microSD slot
expansion connectors	Two standard Pmod ports Arduino/chipKIT Shield connector

4.2. Results

OI-DSCNN has the advantage of less convolution kernel parameters. To evaluate this, CNN was modeled to compare the number of used parameters. The architecture of CNN was shown in Table 5. As Figure 6a displayed, the total amount of convolutional layer parameters between CNN and OI-DSCNN dropped dramatically from 316 to 162 in Pytorch. In Arty Z7, the Winograd algorithm was used, and bias was removed because of the computing acceleration and better performance. OI-DSCNN has a slight increase in the total amount of convolutional layer parameters between the PC and FPGA platforms. The convolutional layer parameters of OI-DSCNN are quantized as 8-bit fixed-point numbers when eventually implemented in FPGA. The OI-DSCNN requires less memory storage resource as shown in Figure 6b. It can be noticed that the convolutional layer parameters of an OIDSCNN requires 1600 bits of memory, where a CNN requires 10,112 bits. The amount of memory used in OI-DSCNN has a significant reduction to 15.8% of the CNN model.



(**a**) Comparison of the number of the convolutional layer parameters.

(**b**) Compare the memory consumption of the convolutional layer parameters.

Figure 6. Convolutional layer parameters for CNN, OI-DSCNN in Pytorch, and OI-DSCNN in Arty Z7.

Table 5. The CNN model architecture.

Layer	Type	Filter Shape	Input Size
Convolution 1	Standard convolution	$10 * 3 * 6$	$10 * 120 * 1$
	Max-pooling	$1 * 2$	$1 * 118 * 6$
Convolution 2	Standard convolution	$1 * 2 * 6 * 10$	$1 * 59 * 6$
	Max-pooling	$1 * 2$	$1 * 58 * 10$
Fully connected 3	Fully connected	$290 * 7$	$1 * 29 * 10$
Classifier	Softmax	-	7

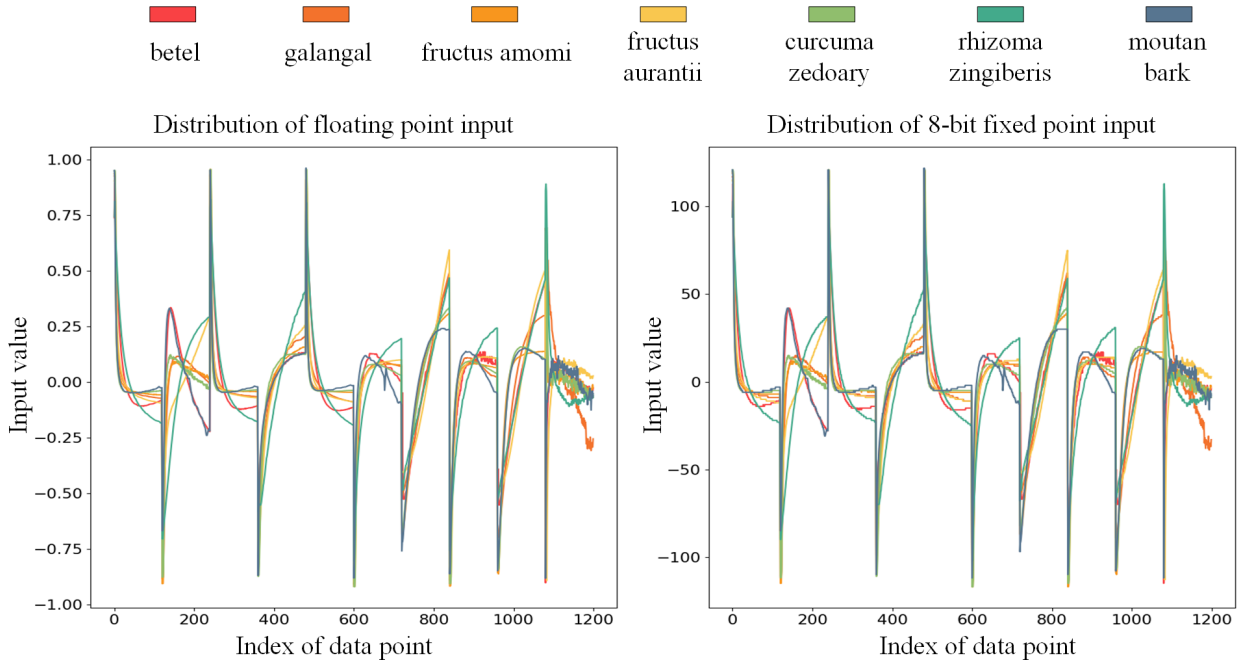
Several algorithms were built for comparison to evaluate the effectiveness of OI-DSCNN. Fivefold cross-validation was used to verify the accuracy of each model in the PC. Simultaneously, OI-DSCNN was quantized and run in Arty Z7 during the cross-validation process to verify the performance of the model on the FPGA. As shown in the Table 6, CNN had the best performance among these models where it took the average score of 0.9457. OI-DSCNN in floating-point calculation had the second-best performance, which was 0.9414, while it had the lowest score of 0.2443 when it operated in 8-bit fixed-point arithmetic. The OI-DSCNN, without bias, operated in 8-bit fixed-point arithmetic had a decent score of 0.9371. Although the accuracy score OI-DSCNN had a narrowed gap

between CNN, CNN+SVM [22], and OI-DSCNN in floating-point arithmetic, OI-DSCNN requires less memory and DSP resources in parallel kernel-operations. Decision tree (DT), multilayer perception (MLP) [46], and principal component analysis with decision tree (PCA+DT) [25] had scores of 0.25, 0.7757, and 0.2686, respectively.

Table 6. Model performance comparison.

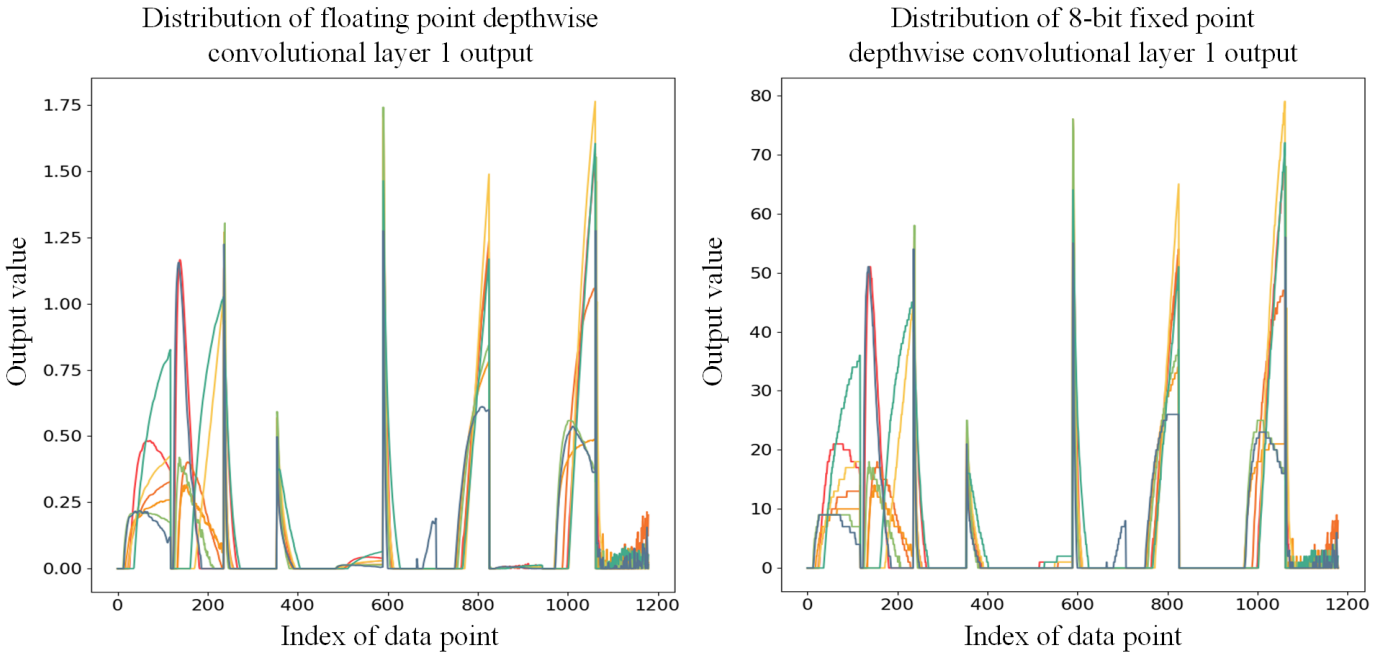
Model	Platform	5-Fold Accuracy					
		1	2	3	4	5	Average
MLP	PC	0.1143	0.2929	0.1643	0.2643	0.4143	0.25
DT	PC	0.65	0.8714	0.7571	0.8071	0.7929	0.7757
PCA + DT	PC	0.3857	0.2929	0.3857	0.1643	0.1143	0.2686
CNN	PC	0.9357	0.95	0.9571	0.9429	0.9429	**0.9457**
CNN + SVM	PC	0.8786	0.8714	0.9214	0.8643	0.8857	0.8843
OI-DSCNN	PC	0.9286	0.9429	0.9643	0.9357	0.9357	0.9414
OI-DSCNN(bias)	Arty Z7	0.2143	0.25	0.2643	0.2357	0.2571	0.2443
OI-DSCNN(nobias)	Arty Z7	0.9286	0.9357	0.9571	0.9357	0.9286	**0.9371**

As the bit width of the output on WDCU, PCU, and FCU blocks are 20-bit, 22-bit, and 22-bit signed fixed points, respectively, the goal of each quantization unit in WDCU, PCU, and FCU is to intercept 8 bits of the result as the final output. Seven samples for each medicine were randomly selected to overview the effectiveness of quantization, as shown in Figure 7. The feature map is reshaped into a one-dimensional matrix for visualization. It can be seen from Figure 7a that the input can maintain a similar distribution after being quantized from floating-point to 8-bit fixed point. It can be noticed that the quantized output of saturation and flooring will losses some information because some lower bits are cut off, and some tiny spikes may be cut off because of the saturation, as shown in Figure 7b–f. However, there is not much difference between the distribution of the actual output and the quantized output distribution. This quantization method does not require any calculations during the implementation period, as long as the saturation cut off and flooring are performed, which is very suitable for FPGAs.

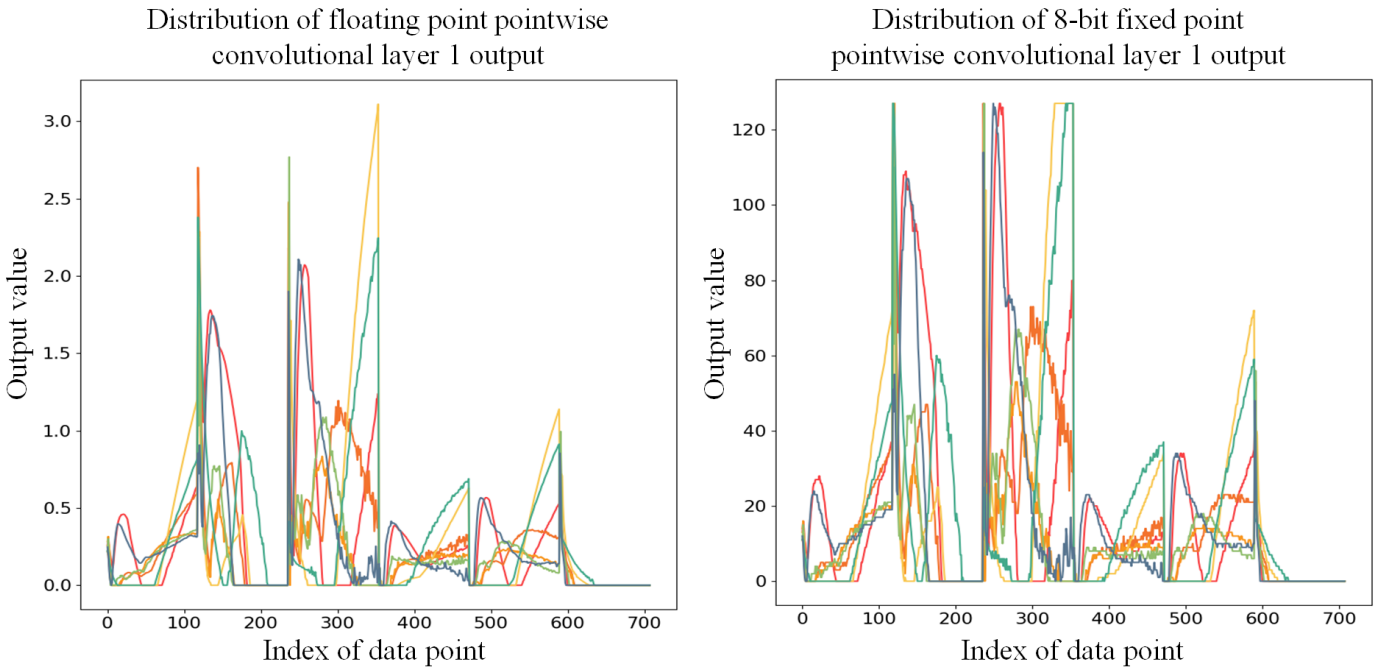


(**a**) Compare distribution of input between floating point and 8-bit fixed point

Figure 7. *Cont.*

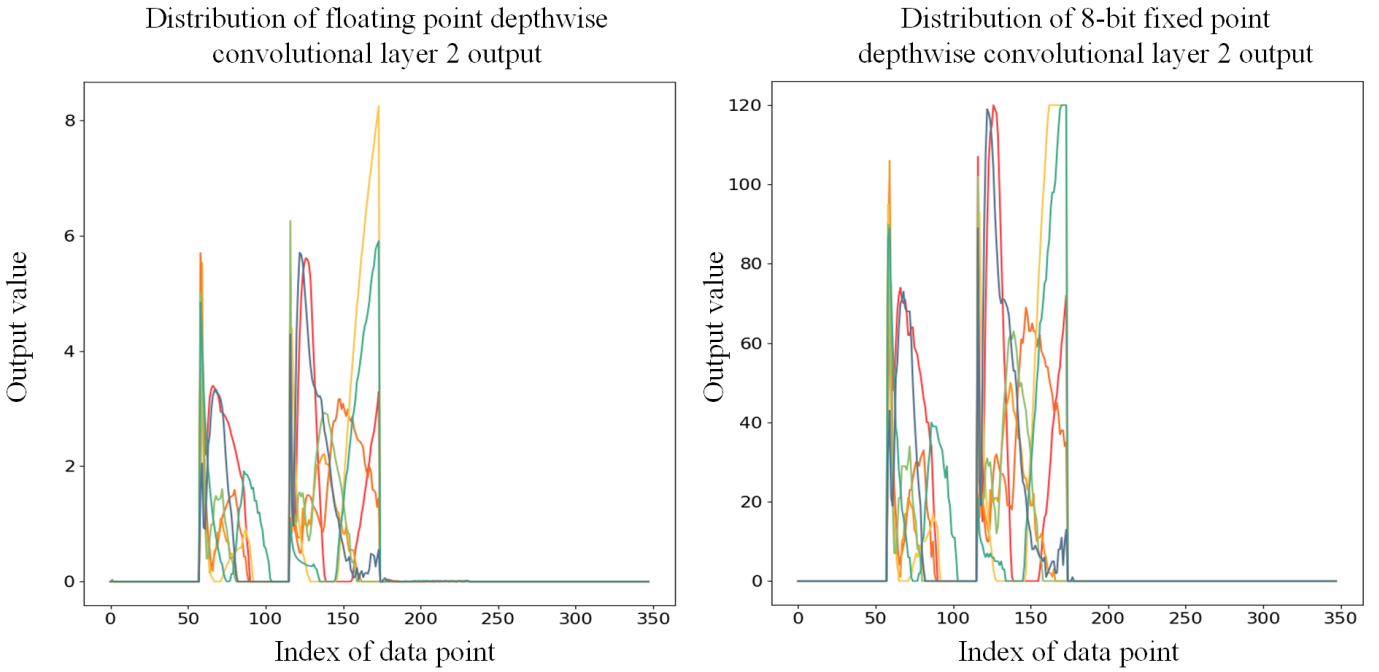


(**b**) Compare distribution of output of depthwise convolutional layer 1 between floating point and 8-bit fixed point

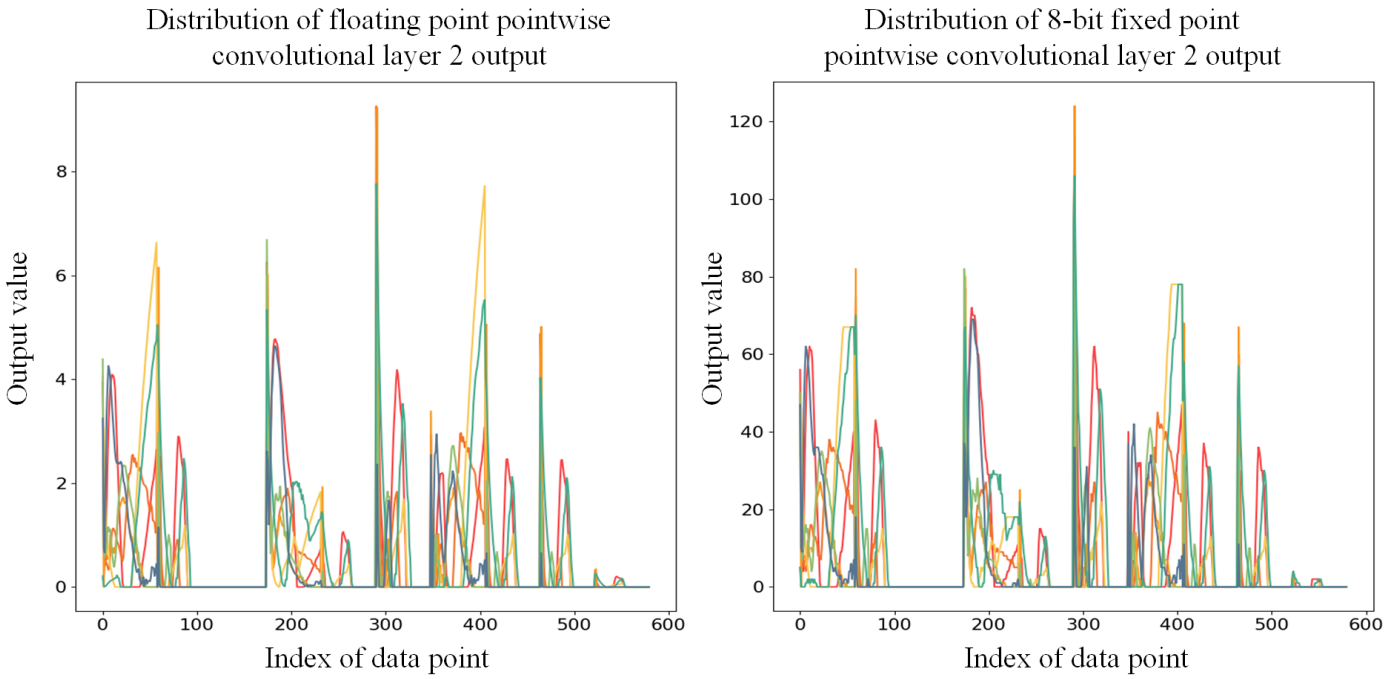


(**c**) Compare distribution of output of pointwise convolutional layer 1 between floating point and 8-bit fixed point

Figure 7. *Cont.*

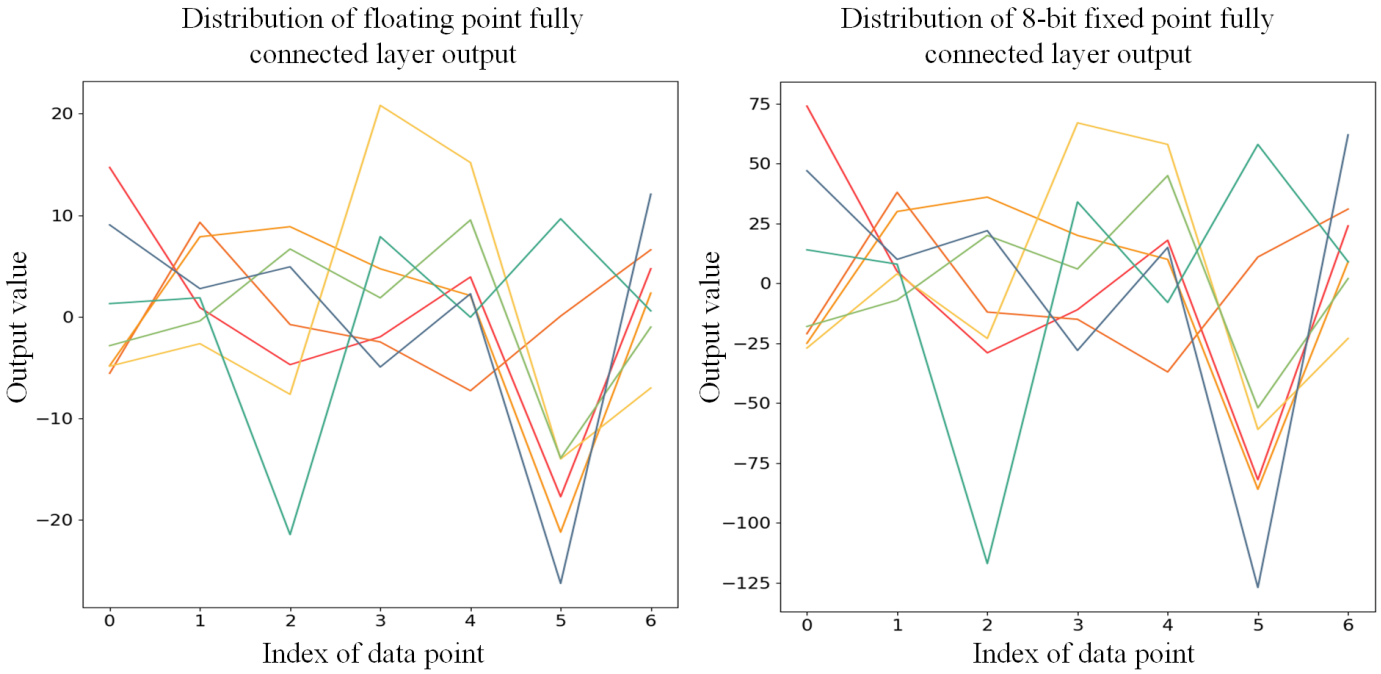


(**d**) Compare distribution of output of depthwise convolutional layer 2 output between floating point and 8-bit fixed point



(**e**) Compare distribution of output of pointwise convolutional layer 2 output between floating point and 8-bit fixed point

Figure 7. *Cont.*



(**f**) Compare distribution of output of fully connected layer between floating point and 8-bit fixed point

Figure 7. The input and output distribution of 7 classes of Chinese medicinal materials in OI-DSCNN. The left side of each subfigure is the distribution of OI-DSCNN at the floating point. The right side of each subfigure is the distribution of OI-DSCNN at an 8-bit fixed point.

OI-DSCNN in 8-bit fixed point operating utilizes less computing resources which can be implemented in some low-cost FPGA platforms. Table 7 demonstrated the resource utilization of OI-DSCNN in the Arty z7-7020 platform. On such a restricted DSP-resource the platform, the algorithm was successfully implemented. A few slice look-up tables (LUTs), registers, and BRAM resources have been consumed, which can be used by the remaining resources to implement other functions.

Table 7. Resource utilization of OI-DSCNN on FPGA.

Type	Slice LUTs	Slice Registers	Block RAM Tile	DSPs
buff	784	1957	9.5	0
Depthwise convolutional layer 1	2305	3164	5	40
Pointwise convolutional layer 1	1309	1878	3	60
Depthwise convolutional layer 2	1434	1980	3	24
Pointwise convolutional layer 2	1396	2130	5	60
Fully connected layer	758	1385	0	35
Utilization	7986	12,494	25.5	219
Avaliable	53,200	106,400	140	220
Percent (%)	15	11.74	18.21	99.5

Finally, a crosswise comparison in commonly used platforms was conducted. Table 8 illustrated the performance comparison of OI-DSCNN on Raspberry Pi 4B which is made in UK Raspberry Pi foundation, CPU, GPU, and Arty z7-7020 platforms. The frequency of the system clock in the Arty Z7-7020 platform was 100MHz. We collected CPU and GPU runtime from PyTorch functions *profiler*. A random sample in the test set was used to test the time for each platform to run OI-DSCNN inference. The model implemented in the Arty platform had a noticeable advance compared to other platforms, where it was 38 times faster than computing in i7-10875H. After that, test set was used to evaluate the

average inference time of each platform. Each odor processing time on Arty Z7 is 20.8 µs, which was better than other platforms. It can be seen that GPU is suitable for training using batch processing models but not for the inference process, especially when the odor data set is small.

Table 8. Comparison between Raspberry Pi, CPU, GPU, and FPGA implementations for OI-DSCNN.

Platform	One Sample Performance		All Sample Performance	
	Time	Speedup	Average Time	Speedup
Raspberry Pi 4B	4.6 ms	0.2×	366.5 µs	0.1×
i7-8750H	6.3 ms	0.3×	57 µs	0.9×
i7-10875H	2.1 ms	1.0×	52 µs	1.0×
NVIDIA GeForce MX250	1107 ms	-	6.325 ms	-
NVIDIA GeForce GTX 2060	910 ms	-	5.2 ms	-
Arty Z7-7020	**57.3 µs**	37×	**20.8 µs**	2.5×

5. Conclusions

In this paper, a lightweight OI-DSCNN is proposed for odor identification. The implementation of separable depthwise convolution and the Winograd algorithm could reduce the number of convolution parameters and accelerate the odor identifying rate. Additionally, the SF-KL is designed to quantize the outputs of each layer of OI-DSCNN and maintain high accuracy. Finally, the OI-DSCNN is successfully implemented in Zynq-7020. The experimental result demonstrates the effectiveness of the system. In summary, OI-DSCNN implemented in FPGA contains fewer parameters and runs faster with higher accuracy. The integration of odor identification algorithms on odor collection devices will become a trend for designing lighter and real-time processing e-noses. Therefore, the integration of e-nose with the balance of performance and cost must be taken into account. Some techniques may be considered to optimize the model, such as better quantization methods and specific parameter reduction strategies. Moreover, FES, a technology that obtains more information from sensors, can be applied to our future research.

Supplementary Materials: The following are available online at https://www.mdpi.com/1424-8220/21/3/832/s1, Section S1: WDCU Design, Section S2: PCU Design, Section S3: FCU Design.

Author Contributions: Conceptualization, Z.M. and D.L.; methodology, Z.M.; software, Z.M.; validation, T.W.; formal analysis, Z.M.; investigation, T.W.; resources, Z.M.; data curation, Z.M.; writing—original draft preparation, Z.M.; writing—review and editing, T.W. and Y.C.; visualization, Z.M. and X.L.; supervision, D.L.; project administration, D.L.; funding acquisition, D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 61571140.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

FPGA	Field-Programmable Gate Array
OI-DSCNN	Odor Identification with Depthwise Separable Convolutional Neural Network
CNN	Convolutional Neural Network
WDCU	Winograd Depthwise Convolution Unit
PCU	Pointwise Convolution Unit
FCU	Fully Connected Unit
KL_divergence	Kullback-Leibler Divergence
DMA	Direct Memory Access
BRAM	Block RAM
DRAM	Distributed RAM
PL	Programmable Logic
PS	Processing System

References

1. Hines, E.; Llobet, E.; Gardner, J. Electronic noses: a review of signal processing techniques. *IEE Proc.-Circuits Dev. Syst.* **1999**, *146*, 297–310. [CrossRef]
2. Fuentes, S.; Summerson, V.; Gonzalez Viejo, C.; Tongson, E.; Lipovetzky, N.; Wilkinson, K.L.; Szeto, C.; Unnithan, R.R. Assessment of Smoke Contamination in Grapevine Berries and Taint in Wines Due to Bushfires Using a Low-Cost E-Nose and an Artificial Intelligence Approach. *Sensors* **2020**, *20*, 5108. [CrossRef] [PubMed]
3. Wei, H.; Gu, Y. A Machine Learning Method for the Detection of Brown Core in the Chinese Pear Variety Huangguan Using a MOS-Based E-Nose. *Sensors* **2020**, *20*, 4499. [CrossRef] [PubMed]
4. Li, W.; Leung, H.; Kwan, C.; Linnell, B.R. E-nose vapor identification based on Dempster–Shafer fusion of multiple classifiers. *IEEE Trans. Instrum. Meas.* **2008**, *57*, 2273–2282. [CrossRef]
5. Kish, L.B.; Smulko, J.; Heszler, P.; Granqvist, C.G. On the sensitivity, selectivity, sensory information and optimal size of resistive chemical sensors. *arXiv* **2007**, arXiv:physics/0701249.
6. Schmera, G.; Kwan, C.; Ajayan, P.; Vajtai, R.; Kish, L.B. Fluctuation-enhanced sensing: Status and perspectives. *IEEE Sens. J.* **2008**, *8*, 714–719. [CrossRef]
7. Kwan, C.; Schmera, G.; Smulko, J.M.; Kish, L.B.; Heszler, P.; Granqvist, C.G. Advanced agent identification with fluctuation-enhanced sensing. *IEEE Sens. J.* **2008**, *8*, 706–713. [CrossRef]
8. Ayhan, B.; Kwan, C.; Zhou, J.; Kish, L.B.; Benkstein, K.D.; Rogers, P.H.; Semancik, S. Fluctuation enhanced sensing (FES) with a nanostructured, semiconducting metal oxide film for gas detection and classification. *Sens. Actuators B Chem.* **2013**, *188*, 651–660. [CrossRef]
9. Aouadi, B.; Zaukuu, J.L.Z.; Vitális, F.; Bodor, Z.; Fehér, O.; Gillay, Z.; Bazar, G.; Kovacs, Z. Historical Evolution and Food Control Achievements of Near Infrared Spectroscopy, Electronic Nose, and Electronic Tongue—Critical Overview. *Sensors* **2020**, *20*, 5479. [CrossRef]
10. Wu, Z.; Zhang, H.; Sun, W.; Lu, N.; Yan, M.; Wu, Y.; Hua, Z.; Fan, S. Development of a Low-Cost Portable Electronic Nose for Cigarette Brands Identification. *Sensors* **2020**, *20*, 4239. [CrossRef]
11. Wu, Y.; Liu, T.; Ling, S.H.; Szymanski, J.; Zhang, W.; Su, S.W. Air quality monitoring for vulnerable groups in residential environments using a multiple hazard gas detector. *Sensors* **2019**, *19*, 362. [CrossRef] [PubMed]
12. Lu, B.; Fu, L.; Nie, B.; Peng, Z.; Liu, H. A Novel Framework with High Diagnostic Sensitivity for Lung Cancer Detection by Electronic Nose. *Sensors* **2019**, *19*, 5333. [CrossRef] [PubMed]
13. Liang, Z.; Tian, F.; Zhang, C.; Yang, L. A Novel Subspace Alignment-Based Interference Suppression Method for the Transfer Caused by Different Sample Carriers in Electronic Nose. *Sensors* **2019**, *19*, 4846. [CrossRef] [PubMed]
14. Wilson, A.D. Application of electronic-nose technologies and VOC-biomarkers for the noninvasive early diagnosis of gastrointestinal diseases. *Sensors* **2018**, *18*, 2613. [CrossRef] [PubMed]
15. Young, R.C.; Buttner, W.J.; Linnell, B.R.; Ramesham, R. Electronic nose for space program applications. *Sens. Actuators B Chem.* **2003**, *93*, 7–16. [CrossRef]
16. Qian, T.; Xu, R.; Kwan, C.; Linnell, B.; Young, R. Toxic vapor classification and concentration estimation for space shuttle and international space station. In *International Symposium on Neural Networks*; Springer: Berlin, Germany, 2004; pp. 543–551.
17. Ryan, M.A.; Zhou, H.; Buehler, M.G.; Manatt, K.S.; Mowrey, V.S.; Jackson, S.P.; Kisor, A.K.; Shevade, A.V.; Homer, M.L. Monitoring space shuttle air quality using the jet propulsion laboratory electronic nose. *IEEE Sens. J.* **2004**, *4*, 337–347. [CrossRef] [PubMed]
18. Ye, S.; Wang, Z.; Shen, J.; Shao, Q.; Fang, H.; Zheng, B.; Younis, A. Sensory qualities, aroma components, and bioactive compounds of Anoectochilus roxburghii (Wall.) Lindl. as affected by different drying methods. *Ind. Crops Prod.* **2019**, *134*, 80–88. [CrossRef]
19. Xu, M.; Yang, S.L.; Peng, W.; Liu, Y.J.; Xie, D.S.; Li, X.Y.; Wu, C.J. A novel method for the discrimination of semen arecae and its processed products by using computer vision, electronic nose, and electronic tongue. *Evid. Based Compl. Alternat. Med.* **2015**, *2015*, 753942. [CrossRef]

20. Wu, D.; Luo, D.; Wong, K.Y.; Hung, K. POP-CNN: Predicting Odor Pleasantness With Convolutional Neural Network. *IEEE Sens. J.* **2019**, *19*, 11337–11345. [CrossRef]
21. Wang, Y.; Diao, J.; Wang, Z.; Zhan, X.; Zhang, B.; Li, N.; Li, G. An optimized Deep Convolutional Neural Network for dendrobium classification based on electronic nose. *Sens. Actuators A Phys.* **2020**, *307*, 111874. [CrossRef]
22. Shi, Y.; Gong, F.; Wang, M.; Liu, J.; Wu, Y.; Men, H. A deep feature mining method of electronic nose sensor data for identifying beer olfactory information. *J. Food Eng.* **2019**, *263*, 437–445. [CrossRef]
23. Roussel, S.; Forsberg, G.; Steinmetz, V.; Grenier, P.; Bellon-Maurel, V. Optimisation of electronic nose measurements. Part I: Methodology of output feature selection. *J. Food Eng.* **1998**, *37*, 207–222. [CrossRef]
24. Carmel, L.; Levy, S.; Lancet, D.; Harel, D. A feature extraction method for chemical sensors in electronic noses. *Sens. Actuators B Chem.* **2003**, *93*, 67–76. [CrossRef]
25. Ali, A.A.S.; Djelouat, H.; Amira, A.; Bensaali, F.; Benammar, M.; Bermak, A. Electronic nose system on the zynq soc platform. *Microprocess. Microsyst.* **2017**, *53*, 145–156. [CrossRef]
26. SIfre, L.; Mallat, S. Rigid-Motion Scattering for Texture Classification. *arXiv* **2014**, arXiv:1403.1687.
27. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
28. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
29. Howard, A.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.C.; Tan, M.; Chu, G.; Vasudevan, V.; Zhu, Y.; Pang, R.; et al. Searching for MobileNetV3. In Proceedings of the IEEE 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–3 November 2019; pp. 1314–1324.
30. Ayhan, T.; Yalçin, M.E. An Application of Small-World Cellular Neural Networks on Odor Classification. *Int. J. Bifurcat. Chaos* **2012**, *22*, 1250013. [CrossRef]
31. Qi, P.F.; Meng, Q.H.; Zeng, M. A CNN-based simplified data processing method for electronic noses. In Proceedings of the 2017 ISOCS/IEEE International Symposium on Olfaction and Electronic Nose (ISOEN), Montreal, QC, Canada, 28–31 May 2017; pp. 1–3.
32. Lekha, S.; Suchetha, M. Real-Time Non-Invasive Detection and Classification of Diabetes Using Modified Convolution Neural Network. *IEEE J. Biomed. Health Inf.* **2018**, *22*, 1630–1636. [CrossRef]
33. Yu, D.; Wang, X.; Liu, H.; Gu, Y. A Multitask Learning Framework for Multi-Property Detection of Wine. *IEEE Access* **2019**, *7*, 123151–123157. [CrossRef]
34. Jong, G.J.; Wang, Z.H.; Hsieh, K.S.; Horng, G.J. A Novel Feature Extraction Method an Electronic Nose for Aroma Classification. *IEEE Sens. J.* **2019**, *19*, 10796–10803. [CrossRef]
35. Xu, S.; Sun, X.; Lu, H.; Zhang, Q. Detection of Type, Blended Ratio, and Mixed Ratio of Pu'er Tea by Using Electronic Nose and Visible/Near Infrared Spectrometer. *Sensors* **2019**, *19*, 2359. [CrossRef] [PubMed]
36. Wang, P.; Hu, Q.; Zhang, Y.; Zhang, C.; Liu, Y.; Cheng, J. Two-Step Quantization for Low-bit Neural Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4376–4384.
37. Park, E.; Ahn, J.; Yoo, S. Weighted-Entropy-Based Quantization for Deep Neural Networks. In Proceedings of the IEEE 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 7197–7205.
38. Ding, W.; Huang, Z.; Huang, Z.; Tian, L.; Wang, H.; Feng, S. Designing efficient accelerator of depthwise separable convolutional neural network on FPGA. *J. Syst. Arch.* **2019**, *97*, 278–286. [CrossRef]
39. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *Computer Vision—ECCV 2016*; Springer: Cham, Switzerland, 2016; pp. 525–542.
40. Liu, Z.; Luo, W.; Wu, B.; Yang, X.; Liu, W.; Cheng, K.T. Bi-Real Net: Binarizing Deep Network Towards Real-Network Performance. *Int. J. Comput. Vis.* **2020**, *128*, 202–219. [CrossRef]
41. Lavin, A.; Gray, S. Fast algorithms for convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4013–4021.
42. Lu, L.; Liang, Y.; Xiao, Q.; Yan, S. Evaluating fast algorithms for convolutional neural networks on FPGAs. In Proceedings of the IEEE 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, CA, USA, 30 April–2 May 2017; pp. 101–108.
43. Li, H.; Fan, X.; Jiao, L.; Cao, W.; Zhou, X.; Wang, L. A high performance FPGA-based accelerator for large-scale convolutional neural networks. In Proceedings of the IEEE 2016 26th International Conference on Field Programmable Logic and Applications (FPL), Lausanne, Switzerland, 29 August–2 September 2016; pp. 1–9.
44. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8026–8037.

45. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
46. Benrekia, F.; Attari, M.; Bouhedda, M. Gas sensors characterization and multilayer perceptron (MLP) hardware implementation for gas identification using a field programmable gate array (FPGA). *Sensors* **2013**, *13*, 2967–2985. [CrossRef] [PubMed]