The Journal of Supercomputing (2024) 80:13176–13186
https://doi.org/10.1007/s11227-024-05947-8

Urban sound classification using neural networks on embedded FPGAs

Jose A. Belloch[1] · Raul Coronado[1] · Oscar Valls[2] · Rocío del Amor[2] · German Leon[3] · Valery Naranjo[2] · Manuel F. Dolz[3] · Adrian Amor-Martin[4] · Gema Piñero[5]

Accepted: 28 January 2024 / Published online: 1 March 2024

Abstract
Sound classification using neural networks has recently produced very accurate results. A large number of different applications use this type of sound classifiers such as controlling and monitoring the type of activity in a city or identifying different types of animals in natural environments. While traditional acoustic processing applications have been developed on high-performance computing platforms equipped with expensive multi-channel audio interfaces, the Internet of Things (IoT) paradigm requires the use of more flexible and energy-efficient systems. Although software-based platforms exist for implementing general-purpose neural networks, they are not optimized for sound classification, wasting energy and computational resources. In this work, we have used FPGAs to develop an ad hoc system where only the hardware needed for our application is synthesized, resulting in faster and more energy-efficient circuits. The results show that our developments are accelerated by a factor of 35 compared to a software-based implementation on a Raspberry Pi.

Keywords FPGA · Sound classification · Hardware acceleration · Convolutional neural networks · Deep learning

1 Introduction

One of the major applications of convolutional neural networks (CNNs) lies in image classification through the use of filters capable of extracting local characteristics in the images [1–3]. Numerous studies have focused on the use of these convolutional networks for applications in speech recognition [4, 5] or music analysis [6, 7]. It was not until 2015 that Piczak created a convolutional neural network specifically for the classification of environmental sounds [8]. Since then, different acoustic features have been studied with the goal of increasing the accuracy of

Extended author information available on the last page of the article

these environmental classification networks. In [9], a network that consisted of two input channels with aggregated features that significantly increased this accuracy was studied. His network was composed of four convolutional layers and one fully connected layer divided into two channels. In addition, it made use of five different acoustic features. The CNN designed in this work is built with the aggregated feature technique as input to the convolutional network. Our contribution is on the implementation of this design from the point of view of performance, which is not commonly addressed in the literature. For example, in [10], an overview of features is shown to take into account for the design, but they do not go deep into efficient implementations of sound classification systems. Our work focuses, on the one hand, on accelerating the execution time of the inference process and, on the other hand, on how the accuracy varies during the quantization process. To this end, we train our model for the purpose of urban sound classification using the aggregated features methodology.

The CNN will be deployed on an Avnet Ultra96-V2 [11] development board using the Vitis AI (VAI) tools from Xilinx [12]. With the VAI development environment, inference times can be accelerated on Xilinx hardware platforms. The use of its Intellectual Property (IP) cores and tools focused on the use of FPGAs in artificial intelligence makes it really easy to port different neural network models to hardware designs in which the inference process can be performed faster and more efficiently than using most microprocessor-based digital systems.

In addition to analyzing the effects of quantization on the percentage accuracy of the model, a system capable of classifying urban sounds individually on the Ultra96-V2 board has been created with the intention of testing which are the limiting elements that affect its application in real time. This system is programmed in the PYNQ framework [13] in order to extract the different acoustic characteristics using Python on the Cortex A-53 microprocessor present on the board.

Finally, an analysis will be performed on the acceleration of spectrogram acquisition using FPGAs in the context of development boards with fewer resources where VAI tools cannot be used for neural network deployment. On these boards, the neural network models can only be used by the embedded processor, leaving the rest of the processes to be accelerated by the FPGA. To perform these tests, the Digilent PYNQ-Z1 development board [14] will be used and a hardware design will be created capable of accelerating the processes necessary for the generation of spectrograms. Figure 1 shows both the two boards used in the development process.

2 Convolutional neural network design

As previously mentioned, our convolutional neural network design is based on [9] with aggregated features. In this paper, five different features were initially considered: Log-Mel Spectrogram, MFCC, Chroma, Spectral Contrast, and Tonnetz. However, we opted not to include the Tonnetz feature due to its significant delay in the feature extraction process. The decision to exclude MFCC was driven by its inherent similarity to the Log-Mel Spectrogram. Specifically, MFCC, which is a compressed representation derived from the Discrete-Cosine Transform (DCT) of the

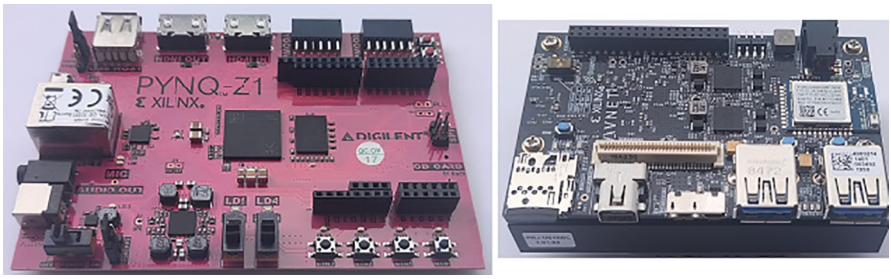

Fig. 1 Digilent PYNQ-Z1 and Avnet Ultra96-V2 are shown on the left and right sides, respectively

Log-Mel Spectrogram, essentially encapsulates similar sound event information. Consequently, to enhance efficiency and avoid redundancy, we focused on the concatenation of the Log-Mel Spectrogram, Chroma, and Spectral Contrast in the first channel. The second channel was dedicated to computing the deltas of the first channel, offering a localized derivative estimation.

As in [9], feature extraction will be done in 41 frames of approximately 23 ms with an overlap of 50% and a sampling rate of 44.1 kHz, resulting in fragments of 943 ms. The bands used for the Log-Mel Spectrogram, Chroma and Spectral Contrast are 60, 12 and 7, respectively. By concatenating these features we have an input size of 41x79. The original model consisted of two different branches for each of the inputs and provided a classification performance of 95%. For the development of this work in the proposed hardware, we decided to simplify the network, using a single branch that receives as the input two different channels, where the second channel is the estimation of the derivative of each feature, which have been extracted using Librosa.

The architecture of the convolutional neural network used is as follows:

1. Input layer of size $41 \times 79 \times 25$.
2. First convolutional layer with 32 kernels of size 3×3, batch-normalization and Rectified Linear Unit (ReLU) activation function.
3. Second convolutional layer with 32 kernels of size 3×3, batch-normalization, ReLU activation and max-pooling with pooling size of 2×2.
4. Third convolutional layer with 64 kernels of size 3×3, batch-normalization, ReLU activation and 2×2 max-pooling.
5. Fourth convolutional layer with 64 kernels of size 3×3, batch-normalization, ReLU activation and 2×2 max-pooling.
6. Fully connected layer with two dense hidden layers of 2048 and 1024 neurons with ReLu activation.
7. Output layer composed of ten units with softmax activation.

To connect the fourth convolutional layer to the fully connected layer we make use of a Flatten layer. With this method, we feed all the resulting data to this last layer. Other techniques were studied such as the use of a Global-Max-Pooling layer that collects the maximum values of each of the feature maps. This greatly increased the

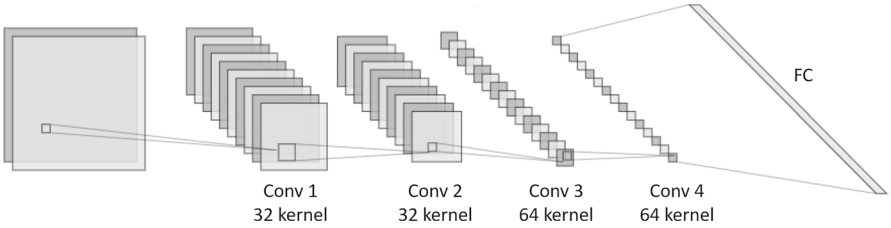

Fig. 2 Architecture of the convolutional neural network

Table 1 Resources used by the DPUCZDX8G in the Avnet Ultra96-V2

Resources	B2304	Available	Percentage (%)
LUT	44,709	70,560	63.36
Register	73,942	141,120	52.39
Block RAM	165	216	76,39
DSP	342	360	95

accuracy of the network, however, such layers are not compatible with the quantization used by the VAI tools. The main analysis of the paper lies in the inference time instead of the accuracy; therefore, we make use of the Flatten layer despite the fact that it implies a degradation in accuracy. Figure 2 shows the structure of the four convolutional layers of the model design.

3 Vitis AI quantization and compilation

For the hardware implementation of the design we use the Xilinx DPUCZDX8G IP, a Deep-learning Processor Unit (DPU) designed for the Zynq UltraScale+ MPSoC [15]. This DPU is optimized for its use with convolutional neural networks. As for its configuration, the B2304 convolution architecture will be selected, which allows 2304 multiply-accumulate operations (MACs) to be performed in each clock cycle. In addition, the low usage mode of digital signal processing (DSP) slices will be used, which means that these slices will only be used in the convolution multiplications and not in the accumulations. Table 1 shows the resources used by the DPU block in the Ultra96-V2.

The original convolutional network design is generated and trained in TensorFlow. After the training process, for its quantization, we will make use of the VAI tools provided by Xilinx. VAI uses data-free quantization through weight equalization and bias correction as its quantization technique [16]. This technique achieves 8-bit quantization with minimal performance degradation. To perform this quantization, the official Xilinx Docker container will be used, which has all the necessary tools to recompile DPU models to be deployed on the board. In this way, from within the docker environment, a script has been created capable of generating, training, quantizing and compiling the models for the desired board. Likewise,

the models are evaluated before and after quantization to compare the difference in accuracy in this same script.

3.1 Performance comparison

To evaluate the model, we will use the UrbanSound8K [17] dataset as a reference, which consists of 8732 urban sound fragments labeled in 10 different classes. We use the tenfold cross-validation method with average accuracy over the 10 splits predefined in the dataset for testing purposes. Table 2 reflects the accuracy of each of the folders before and after the quantization process. Our approach here is to compare the degradation of the performance due to the quantization in a non-optimized implementation such as our basic convolutional network design.

It can be seen from Table 2 that there is no significant degradation in accuracy when quantizing the 32-bit floating-point model to INT8. Moreover, it can be seen that there is even a slight improvement in performance once the model has been quantized for this specific dataset, going from a 60.52% accuracy rate before quantization to 60.96% once it has been quantized. If we look at the performance of the model we can observe which are the folders that achieve the worst accuracy rate, specifically folder three is the split with the worst performance with a 52.40% accuracy rate before quantization and 51.73% after quantization. On the other hand, folder nine achieves the highest performance with 68.54% and 68.55%, before and after the quantization process, respectively.

In order to compile the quantized network we will need a number of files and configuration parameters which includes an “arch.json" file specifying the target board and the convolution architecture that we want to use. In our case, it contains the fingerprint that points to the Ultra96 board and the B2304 architecture. It is important to note that the VAI version used only allows having a single subgraph of the model for the entire DPU block. Functions such as sigmoidal activation are not compatible with quantization and therefore will be scheduled on the CPU. This causes more

Table 2 Performance of the model before and after the quantization process

Folder	Pre-quantization performance	Post-quantization performance
1	0.596768	0.578222
2	0.581106	0.599666
3	0.524094	0.517311
4	0.603009	0.646164
5	0.639304	0.64528
6	0.597834	0.599409
7	0.587644	0.60072
8	0.63193	0.614124
9	0.685424	0.685542
10	0.618819	0.615945
Average	0.605235	0.609604

than one subgraph to be created by having to send information between the DPU and the CPU. These functions cannot be used in this version of VAI and will be replaced by activation functions that can be quantized like ReLU. Once the model is correctly quantized and compiled, we will obtain the neural network in ".xmodel" format that can be used directly by the DPU.

4 Deployment of the model

The model is deployed using the PYNQ framework. To do this we will need to have the PYNQ image on the microSD card from which the board will be booted. We have used PYNQ version 2.7 which is compatible with Xilinx Tools version 2020.2 and uses VAI version 1.4. Once the model is compiled and we have obtained the model in ".xmodel" format it can be directly loaded into the DPU-PYNQ overlay.

The tests performed involved evaluating and labeling all the audio files from each of the splits of the UrbanSound8K dataset in order to test the execution time of the inference process of the new hardware model. For this, we will need to load the DPU hardware design as an overlay folDPSd by the model in ".xmodel" format. Once we have the validation data loaded on the board, we can start making predictions and check the speedup of the hardware model compared to the original model.

To make these predictions we will use the Vitis AI Runtime (VART) which allows the use of the DPU block asynchronously. The inference process of the hardware convolutional neural network model quantized and compiled with VAI takes 2.1 ms for each of the 943 ms fragments. This is a substantial improvement in execution time concerning the tests performed on a general-purpose personal computer (GPC), equipped with an Intel Core i5-7200U processor, and on a Raspberry Pi. The GPC used the original TensorFlow model and the Raspberry Pi is loaded with a TensorFlow Lite quantized model. The execution times in these cases are reflected in Table 3 and are 48.9 ms for the GPC and 85 ms in the case of the Raspberry Pi, giving resulting speedup coefficients of 20.37 and 35.42, respectively.

4.1 Individual audio labeling design

To have a complete system that allows the classification of urban sounds, a program aimed at the Ultra96-V2 was created to classify and label sounds individually. This requires extracting the Log-Mel Spectrogram, Chroma and Spectral Contrast features from the board itself for each of the audio files before they can be classified.

Table 3 Time comparison of the inference process between the Ultra96-V2, PC and Raspberry Pi

	Inference execution time (ms)	Acceleration coefficient (with respect to Ultra96)
Ultra96-V2	2.1	1
GPC	48.9	20.37
Raspberry Pi	85	35.42

With Librosa we can extract the acoustic characteristics of the audio files that serve as input to our hardware model.

Figure 3 represents the flowchart of the program for labeling individual audio files. The inference process is still 2.1 ms, and the feature extraction process takes approximately 250 ms for each 943 ms fragment. By obtaining a processing time less than this threshold of 943 ms, we are fulfilling real-time conditions.

5 Spectrogram hardware acceleration

In the case of development boards with a reduced amount of resources that are not capable of deploying neural network models in hardware using VAI, we propose to speed up the feature extraction process as an alternative. The tests performed during this section are based on the convolutional neural network model proposed by Piczak in 2015 [8], where the Log-Mel Spectrogram is the only feature used as input to the network and will be deployed on the Digilent PYNQ-Z1 development board, specifically designed to make use of the PYNQ framework.

To create a spectrogram from an audio signal, it must first be windowed in order to divide it into different chunks of fixed length. In our case, we will use chunks of 23 ms. Once we have all the fragments, we perform the Fast Fourier Transform (FFT) in each of them to calculate the magnitude of the frequency spectrum of the signal. Each of these spectra represents a small portion of the signal and when we concatenate them all together we obtain the spectrogram.

The hardware design that has been created for the PYNQ-Z1 FPGA is based on the FFT IP from Xilinx. The calculation of this FFT will be performed in hardware, while all the rest of the spectrogram creation process will be performed in software by the Cortex-A9 microprocessor on the board. All the programming of the microprocessor is done through PYNQ. Additional tests are carried out, calculating the FFTs in software to compare the speed in both cases and test the effectiveness of hardware acceleration.

Communication between the processing system (PS) and the programmable logic (PL) is achieved through AXI Direct Memory Access (DMA). The PS will be in charge of fragmenting the audio file into different chunks. The chunks have a size

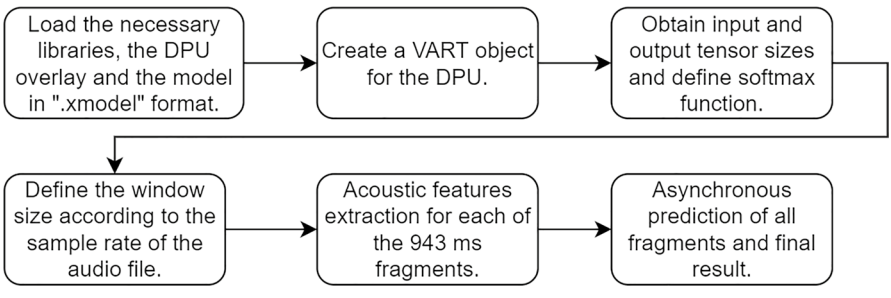

Fig. 3 Flow diagram of the individual labeling program

of 1024 samples since the FFT IP works with that number of samples. Audio files with a sampling rate of 44.1 kHz have been chosen so that the size of each chunk corresponds to the selected duration of 23 ms. In addition, if the audio is in the stereo format it will be converted to mono to facilitate the hardware calculations. Once all the audio chunks are obtained, they will be sent by AXI DMA to the PL one by one, waiting to receive the output back by DMA before sending the next set of data. Finally, the data are interpreted by the PS.

Figure 4 represents the hardware design in Vivado, composed of three main blocks. The Zynq Processing System represents the microprocessor. The AXI DMA IP provides direct access between memory and target peripherals with AXI-Stream inputs. In our design, this block has two functions, converting the input data coming from the microprocessor from AXI to AXI-Stream format to connect it to the input of the FFT block and the reverse process of converting AXI-Stream data from the output of the block to AXI format to be sent back to the microprocessor. The data sending in both cases is done through a High Performance (HP) port. Finally, the Fast Fourier Transform block performs the calculation of the different FFTs.

In order to assess the portability of our implementation, we carried out tests by using synthetic audio files of different lengths so that we can check our design operates at reasonable processing times. To this end, we generate audio files varying from 50 ms to 10 s and test the rate the data were transferred to the board. The results indicated that the processing time is proportional to the data.

6 Conclusion

In this work, we have tested the capabilities of FPGAs in the acceleration of artificial intelligence applications, specifically focused on convolutional neural networks for urban sound classification. By using the Xilinx tools, it has been possible to create a hardware neural network model capable of performing the inference process more efficiently in comparison with microprocessor-based systems. In addition, it has been proven that the quantization process of the model does not produce a significant degradation in the percentage of accuracy of the network, even being able to increase it slightly. For development boards that do not have enough resources to implement and deploy these hardware models, it has been proposed to accelerate the

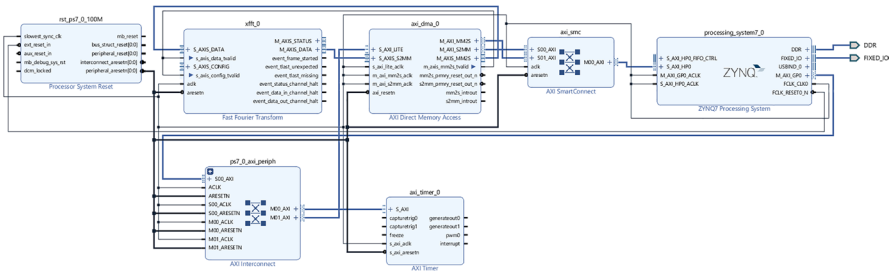

Fig. 4 Vivado block design for FFT calculation

extraction of spectrograms using FPGAs. In this way, two different approaches have demonstrated to take advantage of the potential that these devices have when working in the field of artificial intelligence.

Author contributions All authors contributed equally to this work.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work has been funded by the NextGenerationEU/PRTR, MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe" through Grants PID2020-113656RB, PID2021-124280OB-C21, PID2022-137048OA-C43, TED2021-131401B-C21 (DIPSY-AI), and TED2021-131401A-C22 (DIPSY-TECH), and by CM through PROGRAMA MIMACUHSPACE-CM-UC3M. G. Piñero and V. Naranjo's work has been partially funded by GVA through PROGRAMA PROMETEO 2023-CIPROM/2022/20. Manuel F. Dolz was also supported by the Plan Gen–T grant CIDEXG/2022/013 of the Generalitat Valenciana.

Data availability No additional data or materials available.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Sultana F, Sufian A, Dutta P (2018) Advancements in image classification using convolutional neural network. In: 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), pp 122–129. https://doi.org/10.1109/ICRCICN.2018.8718718
2. Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput 29(9):2352–2449. https://doi.org/10.1162/neco_a_00990
3. Sharma N, Jain V, Mishra A (2018) An analysis of convolutional neural networks for image classification. Procedia Comput Sci 132:377–384. https://doi.org/10.1016/j.procs.2018.05.198
4. Richardson F, Reynolds D, Dehak N (2015) Deep neural network approaches to speaker and language recognition. IEEE Signal Process Lett 22:1–1. https://doi.org/10.1109/LSP.2015.2420092
5. Ali H, Tran S, Benetos E, Garcez A (2018) Speaker recognition with hybrid features from a deep belief network. Neural Comput Appl. https://doi.org/10.1007/s00521-016-2501-7
6. Dieleman S, Brakel P, Schrauwen B (2011) Audio-based music classification with a pretrained convolutional network, pp 669–674
7. Ghosal D, Kolekar M (2018) Music genre recognition using deep neural networks and transfer learning, pp 2087–2091. https://doi.org/10.21437/Interspeech.2018-2045
8. Piczak KJ (2015) Environmental sound classification with convolutional neural networks. In: 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), pp 1–6. https://doi.org/10.1109/MLSP.2015.7324337

9. Su Y, Zhang K, Wang J, Madani K (2019) Environment sound classification using a two-stream CNN based on decision-level fusion. Sensors 19:1733. https://doi.org/10.3390/s19071733
10. Vandendriessche J, Wouters N, da Silva B, Lamrini M, Chkouri MY, Touhafi A (2021) Environmental sound recognition on embedded systems: from fpgas to tpus. Electronics 10(21):2622
11. Ultra96-V2 Single Board Computer Hardware User's Guide. https://www.96boards.org/documentation/consumer/ultra96/ultra96-v2/hardware-docs/. Accessed 25 Oct 2022
12. Vitis AI User Guide (UG1414). https://docs.xilinx.com/r/2.0-English/ug1414-vitis-ai. Accessed 25 Oct 2022
13. PYNQ - Python productivity for Zynq. http://www.pynq.io/. Accessed 25 Oct 2022
14. PYNQ-Z1 Reference Manual. https://digilent.com/reference/programmable-logic/pynq-z1/reference-manual. Accessed 25 Oct 2022
15. DPUCZDX8G for Zynq UltraScale+MPSoCs Product Guide (PG338). https://docs.xilinx.com/r/en-US/pg338-dpu. Accessed 25 Oct 2022
16. Nagel M. Baalen MV, Blankevoort T, Welling M (2019) Data-free quantization through weight equalization and bias correction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)
17. Salamon J, Jacoby C, Bello JP (2014) A dataset and taxonomy for urban sound research. In: 22nd ACM International Conference on Multimedia (ACM-MM'14), Orlando, FL, USA, pp 1041–1044

Authors and Affiliations

Jose A. Belloch[1] · Raul Coronado[1] · Oscar Valls[2] · Rocío del Amor[2] · German Leon[3] · Valery Naranjo[2] · Manuel F. Dolz[3] · Adrian Amor-Martin[4] · Gema Piñero[5]

✉ Jose A. Belloch
jbelloc@ing.uc3m.es

Raul Coronado
rcoronad@ing.uc3m.es

Oscar Valls
osvallo@i3b.upv.es

Rocío del Amor
madeam2@upvnet.upv.es

German Leon
leon@uji.es

Valery Naranjo
vnaranjo@dcom.upv.es

Manuel F. Dolz
dolzm@uji.es

Adrian Amor-Martin
aamor@ing.uc3m.es

Gema Piñero
gpinyero@iteam.upv.es

[1] Depto. de Tecnología Electrónica, Universidad Carlos III de Madrid, Avda Universidad 30,

28911 Leganés, Madrid, Spain

[2] Instituto Universitario de Investigación en Tecnología Centrada en el Ser Humano, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Valencia, Spain

[3] Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I de Castellón, Avda. Sos Baynat s/n, 12071 Castellón, Castellón, Spain

[4] Depto. Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, Avda Universidad 30, 28911 Leganés, Madrid, Spain

[5] Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politecnica de Valencia, Camino de Vera s/n, 46022 Valencia, Valencia, Spain