A Computational Architecture for Inference of a Quantized-CNN for Detecting Atrial Fibrillation

Andrés F. Jaramillo Rueda[1], Laura Y. Vargas Pacheco[2]and Carlos A. Fajardo[3]

Abstract

Atrial Fibrillation is a common cardiac arrhythmia, which is characterized by an abnormal heartbeat rhythm that can be life-threatening. Recently, researchers have proposed several Convolutional Neural Networks (CNNs) to detect Atrial Fibrillation. CNNs have high requirements on computing and memory resources, which usually demand the use of High Performance Computing (eg, GPUs). This high energy demand is a challenge for portable devices. Therefore, efficient hardware implementations are required. We propose a computational architecture for the inference of a Quantized Convolutional Neural Network (Q-CNN) that allows the detection of the Atrial Fibrillation (AF). The architecture exploits data-level parallelism by incorporating SIMD-based vector units, which is optimized in terms of computation and storage and also optimized to perform both the convolutional and fully connected layers. The computational architecture was implemented and tested in a Xilinx Artix-7 FPGA. We present

---

[1] Universidad Industrial de Santander, af_jaramillo@outlook.com , Bucaramanga, Colombia.

[2] Universidad Industrial de Santander, lauvapacheco@gmail.com, Bucaramanga, Colombia.

[3] Universidad Industrial de Santander, cafajar@uis.edu.co, Bucaramanga, Colombia.

the experimental results regarding the quantization process in a different
number of bits, hardware resources, and precision. The results show an accuracy of 94% accuracy for 22-bits. This work aims to be the basis for the
future implementation of a portable, low-cost, and high-reliability device
for the diagnosis of Atrial Fibrillation.

Keywords: Atrial fibrillation; automatic detection; FPGA
implementation; quantized convolutional neural network.

Arquitectura Computacional para la Inferencia de una CNN Cuantizada para Detectar Fibrilación Auricular

Resumen

La fibrilación auricular es una arritmia cardíaca común, que se caracteriza por un ritmo cardíaco anormal que puede poner en peligro la vida.
Recientemente, se han propuesto varias Redes Neuronales Convolucionales (CNNs, por sus siglas en inglés) para detectar la fibrilación auricular.
Las CNN tienen altos requisitos de recursos informáticos y de memoria,
lo que generalmente demanda el uso Computación de Altro Rendimiento
como por ejemplo GPUs. Esta alta demanda de energía es un desafío para los dispositivos portátiles. Por lo tanto, se requieren implementaciones
de hardware eficientes. Proponemos una arquitectura computacional para la inferencia de una Red Neural Convolucional Cuantizada (Q-CNN)
que permite la detección de la Fibrilación Auricular (FA). La arquitectura
aprovecha el paralelismo a nivel de datos, incorporando unidades vectoriales basadas en SIMD, que están optimizadas en términos de cálculo
y almacenamiento. El diseño también se optimizó para realizar tanto las
capas convolucionales como las capas completamente conectadas. La arquitectura computacional se implementó y probó en una FPGA Xilinx
Artix-7. Presentamos los resultados experimentales con respecto al proceso de cuantización en un número diferente de bits, recursos de hardware
y precisión. Los resultados muestran una precisión del 94 % para 22 bits.
Este trabajo pretende ser la base para la futura implementación de un
dispositivo portátil, de bajo costo y alta confiabilidad para el diagnóstico
de Fibrilación Auricular.

Palabras clave: Detección automática; fibrilación auricular;
implementación en FPGA; red neuronal convolucional cuantizada.

1 Introduction

Atrial fibrillation (AF) is an arrhythmia that presents irregular heartbeats,
and it is associated with an increase in heart rate due to a disorder in

the electrical signals that activate the atria. This type of arrhythmia occurs asymptomatically, to say, there are no symptoms until the first acute episode [1]. However, it is difficult to accurately detect AF in the early stage, and well-trained professional physicians are required to accurately determine the feature information of ECG [2]. Therefore, it is important to develop fast and accurate algorithms for AF automatic detection.

To address this challenge, several studies have proposed the convolutional neural networks (CNN) for the detection of atrial fibrillation with high levels of accuracy [2],[3],[4],[5]. Moreover, some researches have shown that custom hardware for the inference of CNNs could surpass the efficiency of general-purpose processor equivalents in terms of throughput and energy consumption [6].

Quantization is an effective strategy that reduces the precision of both weights and activations. The quantization of a CNN is the first step before implementing a CNN in a custom-hardware.

FPGAs have become striking to implement Q-CNNs because of their flexibility and high energy efficiency. These versatile integrated circuits provide programmable logic blocks and a configurable interconnection, which enable the construction of custom accelerator architectures in the custom hardware [7]. However, there are still many challenges because the CNNs are known for demanding a massive amount of computational and memory resources.

Strategies to perform the inference process *at the edge* are currently a hot topic in hardware researches. The authors in [8] propose a specific dataflow to minimize the memory access and data movement while maximizing the resource utilization. In [9] is proposed a Winograd transformation-based algorithm to optimize the convolution process, which uses a cross-layer strategy. The algorithm allows a reduction of over 90% in the transfer process of the intermediate data. The authors in [10] proposes an accelerator to handle network layers of different scales through parameter configuration and maximizes bandwidth by using a data stream interface. In [11] is proposed a reconfigurable CNN accelerator that reduces the number of off-chip memory accesses by combining convolution and pooling operations and using a 16-bit dynamic fixed-point format. For further details in custom hardware accelerators, the readers may refer to

recent surveys on this topic in [12],[13]. The first one ([12]) focuses on custom hardware in general for CNNs. The second one ([13]) focuses on FPGA-based accelerators for CNNs.

In this work, we propose a computational architecture for the inference process of a quantized version of the Castillo-Granados CNN [14]. Our goal is to design a specific purpose processor that carries out the inference process by using the minimum amount of computational and memory resources at high accuracy possible. We designed a SIMD architecture (Single Instruction, Multiple Data) with a single vector unit that is optimized to perform both the convolution and fully connected layers. This processor allows the inference of a 22-bits Q-CNN version [14] and achieves a 94% accuracy.

This paper is organized as follows: Section II gives a description of the CNN used. Section III describes the quantization process of CNN. Section IV describes the design of the computational architecture. Section V summarizes the main results of this work. Finally, the article is closed with the conclusions in Section VI.

2   Convolutional neural network

A typical CNN is made up of different layers. In each layer, there is a certain amount of connected filters that extract information for subsequent layers. The input data passes through layers to generate a feature vector. Then, a classifier is used in the characteristic vector obtained to produce the result of the classification. There are mainly three types of layers in a CNN model: convolutional layers, grouping layers, and fully connected (FC) layers.

In this paper, the CNN Castillo-Granados [14] is implemented (Figure 1). This model was trained for the detection of AF from ECG signals. These ECG signals were registered by the Einthoven triangle method [15] and stored in a vector of 500 samples with a sampling rate of 250 [samples/s]. This CNN achieved an accuracy of 97.44% using a 64-bit double-float format [14].
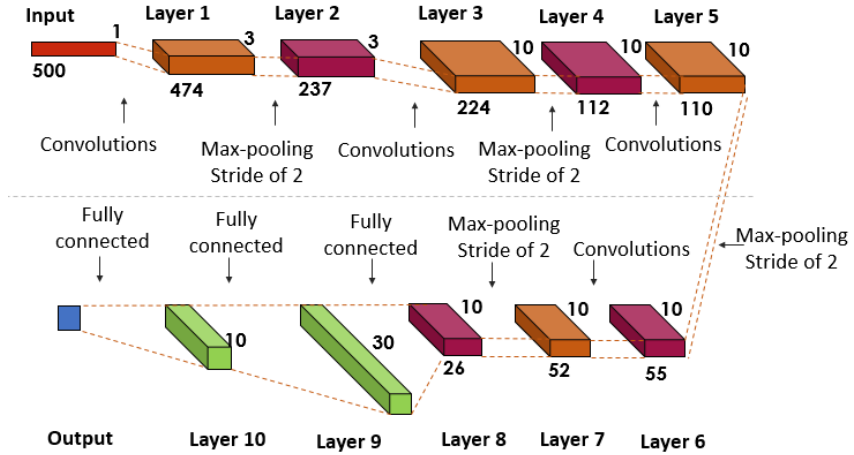
Figure 1: Castillo-Grandos CNN Architecture [14]

The CNN has four convolution layers followed by three FC layers. Table 1 summarizes the characteristics of the layers.

Table 1: Layers description: Layer type, Output dimensions, Number of parameters, Kenel Size and number of kernels (#) and stride used. (Adapted from [14])

Neural Network Architecture				
Layer type	Output dim.	Parameters	K. size, #	Stride
Input	(500.1)	-	-	-
Convolution	(474.3)	84	(27.3)	1
Max-pooling	(237.3)	-	-	2
Convolution	(224.10)	430	(14.10)	1
Max-pooling	(112.10)	-	-	2
Convolution	(110.10)	310	(3.10)	1
Max-pooling	(55.10)	-	-	2
Convolution	(52.10)	410	(4.10)	1
Max-pooling	(26.10)	-	-	2
Flatten	260	-	-	-
Fully-connected	30	7830	-	-
Fully-connected	10	310	-	-
Fully-connected	1	11	-	-

The network has a total of 9385 parameters and 377428 fixed-point operations (additions and multiplications). Figures 2a and 2b show the distribution percentages of the number of operations carried out and the number of parameters required in both the convolutional and FC layers. Note that, on the one hand, the convolutional layers perform the highest percentage of operations (96% vs. 4%). On the other hand, the FC layers require the highest percentage of parameters (87% vs. 13%).
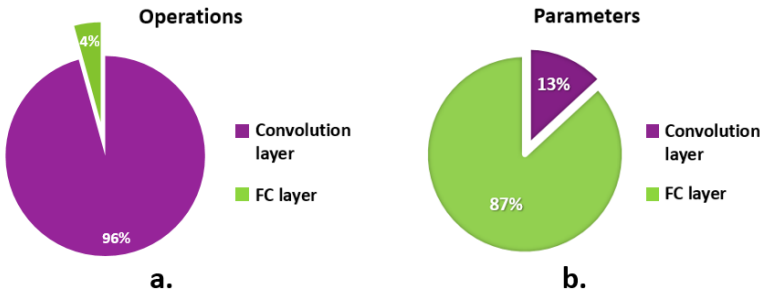

Figure 2: Percentage of operations and parameters for convolutional and fully connected layers.

3 Quantization process

The implementation of the inference process in custom hardware requires a quantization process. This process allows us to change 64-bit floating-point format for a reduced number of bits by using a fixed-point format. This change reduces considerably the amount of computational and memory resources. Figure 3 shows the results of the *fake quantization* process that was carried out using Matlab. Note that by using just 12 bits an accuracy of 95% is achieved. Also, note that from 12 bits onwards there is no considerable increase in the accuracy. However, in the hardware implementation, there are some issues related to the truncation error because of the reduction in the number of bits, which will be analyzed in Section 5.
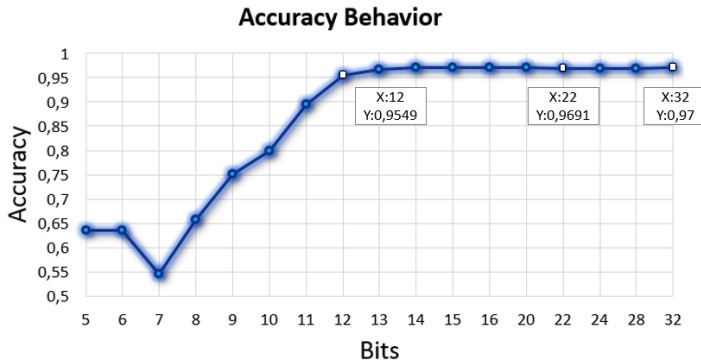
Figure 3: Relationship between the number of bits and the accuracy

4   Design description

Figure 4 shows a block diagram of the computational architecture designed for the inference of the CNN of Figure 1.
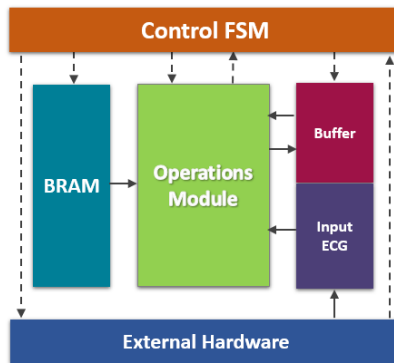

Figure 4: Block diagram of computational architecture. **Dashed line:* **Control signal. Continuous line:**Data

The design has an operation module that computes the input data with the parameters of each layer. This module is controlled by a Finite State Machine (FSM control) that addresses the computational resources that carry out the mathematical operations.

We have designed a computational strategy that allows us to use a

single *operation module* to perform both the convolutional and FC layers.
This strategy demands the use of buffers to temporarily store output re-
sults. Thus, the proposed architecture achieves a considerable reduction in
the use of computational resources. However, this strategy penalizes the
throughput because the reuse strategy does not allow a pipeline implemen-
tation.

A functional description of the modules in Figure 4 is given below :

- *Control FSM*: State machine addresses the flow of data processed in
  each module. Also, the FSM controls the *Operation Module* to per-
  form all layers. Finally, the FMS carries out the write/read memory
  process.

- *BRAM*: Memory to store all parameters of the CNN.

- *Operations Module*: adaptive module that computes convolution or
  FC operations.

- *Buffer*: Set of two memories that store the temporary outputs of each
  layer, alternating writing, and reading, the read data is returned as
  inputs of the next layer.

- *Input ECG*: Memories that stores the ECG segments. In this design,
  there are two Input ECG memories, which allow us to read a new
  segment while a previous segment is being processing.

- *External Hardware*: ECG signals are acquired through External Hard-
  ware. This module communicates the FPGA with the ADC using the
  SPI protocol. *External hardware* provides the data to *Input ECG* in
  groups of 500 samples with a sampling frequency of 250 [samples/s].

4.1   Design of the operations module

The operations processing module has been designed based on the convo-
lution layers since these contain the largest number of operations in the
architecture (Figure 2).

The *Operations Module* uses a loop unrolling strategy for the kernels
in the convolution layers [16]. A SIMD-based architecture carries out this

strategy by a sliding buffer, which contains 27 multipliers, 27 adders, and 27 shift registers. This custom processor allows the reuse of the hardware for all layers.
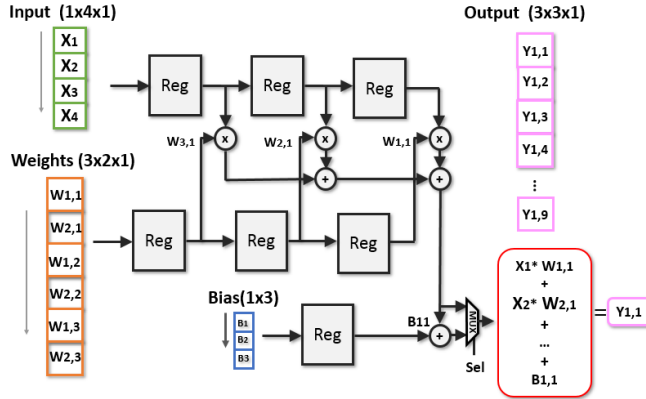

Figure 5: Data flow in *Operations Module*

Figure 5 illustrates the configuration of the logistic resources used for the execution of operations. Note that Kernel and input data (Section 2) flow from left to right in each clock cycle until all 27 registers are filled. Once the first 27 data are saved on the registers, a first temporary data out is obtained. Then, the input data is 1-left shifted and a second temporary data is obtained, and so on. All temporary data are accumulated in a specific position of the Buffer memory.

It is important to note that the dimensions change from one layer to another, so the bias is added in the last tensor dimension.

The FSM controls the data flow by modifying the control signals. The design can be configured to calculate both convolutional and FC layers. This strategy saves the use of logical resources for the description of layers that execute different operations. If better latency is required, more parallelism can be applied (more than 27 operations per clock cycle) and more than one kernel at a time.

4.2   Hard-limit transfer function

The original design of the neural network was performed with the *Sigmoid*
activation function [17] (Figure 6). This function is applied after the last
FC layer. We replaced the *Sigmoid* function with a Hard-limit function to
reduce computational resources. This function was implemented by using
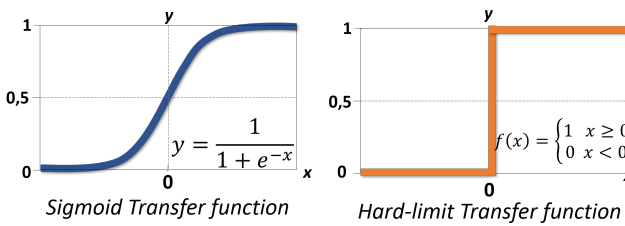a simple not gate.

Figure 6: Sigmiod function and Hard-limit function

Our results suggest that the use of a *Hard-limit* function does not affect
the accuracy of the network.

5   Results

The computational architecture was implemented on the Basys 3 Deve-
lopment Board which is based on the latest Artix-7 FPGA from Xilinx.
The synthesis, simulation, and debugging was carried out using the Xilinx
Vivado Design Suite ® software with the 2019.1 version.

The design was tested with a set of 1000 ECG signals of the MIT BIH
Atrial Fibrillation database [18]. These signals were quantized from 12 to 32
bits by using Matlab (Section 3). Several tests were developed to validate
intermediate and final results. The intermediate results were validated by
ILA Tool from Vivado ®.

The percentage truncation error (E_t) is generated for the reduction in
the number of bits and calculated by Equation 1.

$$E_t = \left| \frac{CHR - SR}{SR} \right| \times 100\% \tag{1}$$

Where CHR is the Custom Hardware Result and SR is Software Result (Matlab). The E_t depends on the number of bits. The error increases when the number the bits is reduced. Besides, this error is propagated through all layers. Thus the bigger E_t is found in the last layer. For example, the E_t, for 22 bits, in the last layer was around 0.79%.

5.1   Hardware resource utilization

Figure 7 shows the percentages (concerning the total available in the FPGA) of resource utilization for a different number of bits. It can be observed that between 12 and 22 bits there is no change in the percentage of DSP utilization.
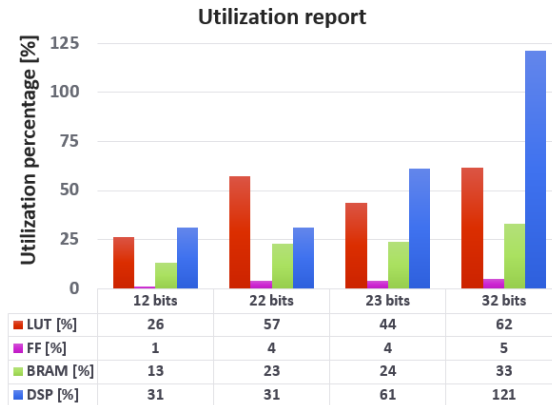

Figure 7: Percentage of utilization for different amounts of bits of quantization

5.2   Accuracy regarding the number of bits

A test was performed using the set of 1000 ECG signals, which 500 corresponds to *Fibrillation signals* and the other 500 with *Not fibrillated signals*. Table 2 summarizes the results.

Table 2: Accuracy on the inferences process for 12 and 22 quantization bits

Bit quantity	Accuracy
12-bits	88 [%]
22-bits	94 [%]

Note that for 12 bits there is an important reduction in the accuracy, which is due to the truncation error. Taking into account the accuracy and the amount of resources required, a 22-bits quantization is adopted.

5.3 Performance

A clock frequency of 34.6 [KHz] was implemented, satisfying the required throughput of an inference every two seconds. Table 3 summarizes the main results to obtain maximum performance on FPGA.

Table 3: Feature performance summary

Feature performance	
Throughput required	0.5 [inferences/s]
Maximum clock frequency (MCF)	25,5 [MHz]
Latency at MCF	1,358 [ms]
Throughput at MCF	736 [inferences/s]

Figure 8 shows the breakdown of the execution time to calculate each CNN layer. Note that convolution operations are the ones that consume the most time, therefore, if better latency is required, parallelism techniques can be applied.
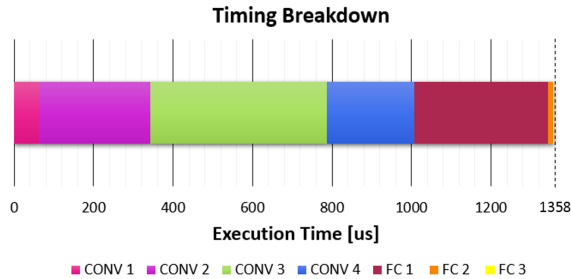
Figure 8: Timing breakdown for the inference process

6 Conclusions

A computational architecture was proposed to carry out the inference process of a Q-CNN, which allows the detection of Atrial Fibrillation.

The design is an architecture SIMD-based vector unit with a sliding buffer that is optimized for both convolutional and FC layers. The design aims to reduce the amount of computational and memory resources. The architecture has a throughput of an inference every two seconds, i.e. it works at 34.6 [KHz]. However, the design can achieve a throughput of 736 [inferences/s] at its is maximum design frequency (25.5[Mhz]). The tests show accuracy in the inference of 94% for 22-bits of quantization, which moves approximately 2.97% away from the inference in 64-bits software. Future work focuses on the use of aware quantization strategies, which can improve accuracy by using a lower amount of bits [19],[20]. We also will test different approximation strategies, which have also proved to improve the accuracy [21]. We aim to use this design on the implementation of a Q-CNN-based portable device for automatic detection of AF.

References

[1] S. S. Chugh, R. Havmoeller, K. Narayanan, D. Singh, M. Rienstra, E. J. Benjamin, R. F. Gillum, Y.-H. Kim, J. H. McAnulty Jr, Z.-J. Zheng *et al.*, "Worldwide epidemiology of atrial fibrillation: a global burden of disease 2010 study," *Circulation*, vol. 129, no. 8, pp. 837–847, 2014. https://doi.org/10.1161/CIRCULATIONAHA.113.005119 137

[2] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, no. 1, pp. 65–69, 2019. http://dx.doi.org/10.1038/s41591-018-0268-3 137

[3] P. A. Warrick and M. Nabhan Homsi, "Ensembling convolutional and long short-term memory networks for electrocardiogram arrhythmia detection," *Physiological Measurement*, vol. 39, no. 11, 2018. http://dx.doi.org/10.1088/1361-6579/aad386 137

[4] Z. Xiong, M. P. Nash, E. Cheng, V. V. Fedorov, M. K. Stiles, and J. Zhao, "ECG signal classification for the detection of cardiac arrhythmias using a convolutional recurrent neural network," *Physiological Measurement*, vol. 39, no. 9, 2018. http://dx.doi.org/10.1088/1361-6579/aad9ed 137

[5] T. Mahmud, S. A. Fattah, and M. Saquib, "Deeparrnet: An efficient deep cnn architecture for automatic arrhythmia detection and classification from denoised ecg beats," *IEEE Access*, vol. 8, pp. 104 788–104 800, 2020. http://dx.doi.org/10.1109/ACCESS.2020.2998788 137

[6] Song Han, "Efficient methods and hardware for deep learning," Ph.D. dissertation, Stanford University, 2017. 137

[7] D. Gschwend, "Zynqnet: An fpga-accelerated embedded convolutional neural network," 2020. https://arxiv.org/pdf/2005.06892.pdf 137

[8] Y. Ma, Y. Cao, S. Vrudhula, and J. S. Seo, "Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks," *FPGA 2017 - Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 45–54, 2017. https://doi.org/10.1145/3020078.3021736 137

[9] J. Yu, Y. Hu, X. Ning, J. Qiu, K. Guo, Y. Wang, and H. Yang, "Instruction driven cross-layer CNN accelerator with winograd transformation on FPGA," *2017 International Conference on Field-Programmable Technology, ICFPT 2017*, vol. 2018-Janua, pp. 227–230, 2018. https://doi.org/10.1109/FPT.2017.8280147 137

[10] S. Sağlam, F. Tat, and S. Bayar, "Fpga implementation of cnn algorithm for detecting malaria diseased blood cells," in *2019 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*. IEEE, 2019, pp. 1–5. http://dx.doi.org/10.1109/ISAECT47714.2019.9069724 137

[11] S. Zhang, J. Cao, Q. Zhang, Q. Zhang, Y. Zhang, and Y. Wang, "An fpga-based reconfigurable cnn accelerator for yolo," in *2020 IEEE 3rd*

International Conference on Electronics Technology (ICET). IEEE, 2020, pp. 74–78. http://dx.doi.org/10.1109/ICET49382.2020.9119500 137

[12] E. Wang, J. J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P. Y. Cheung, and G. A. Constantinides, "Deep neural network approximation for custom hardware: Where we've been, where we're going," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–39, 2019. https://doi.org/10.1145/3309551 138

[13] S. Mittal, "A survey of fpga-based accelerators for convolutional neural networks," *Neural computing and applications*, vol. 32, pp. 1109–1139, 2020. https://doi.org/10.1007/s00521-018-3761-1 138

[14] J. A. Castillo, Y. C. Granados, and C. A. Fajardo, "Patient-Specific Detection of Atrial Fibrillation in Segments of ECG Signals using Deep Neural Networks," *Ciencia E Ingenieria Neogranadina*, vol. 30, no. 1, 2020. https://doi.org/10.18359/rcin.4156 138, 139

[15] W. Uribe, M. Duque, and E. Medina, "Electrocardiografía y arritmias," *Clínica Medellín. Editorial PLA Export Bogotá DC*, pp. 41–5, 2005. 138

[16] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks." EPFL, Aug 2016, pp. 1–9. https://doi.org/10.1109/FPL.2016.7577308 142

[17] A. Tisan, S. Oniga, D. MIC, and A. Buchman, "Digital Implementation of The Sigmoid Function for FPGA Circuits," *Acta Technica Napocensis*, vol. 50, no. 2, pp. 15–20, 2009. 144

[18] A. Goldberger, "MIT-BIH atrial fibrillation database," 2000. https://physionet.org/content/afdb/1.0.0/ 144

[19] 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," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018. https://doi.org/10.1109/CVPR.2018.00286 147

[20] Y. Dong, R. Ni, J. Li, Y. Chen, H. Su, and J. Zhu, "Stochastic Quantization for Learning Accurate Low-Bit Deep Neural Networks," *International Journal of Computer Vision*, vol. 127, no. 11, pp. 1629–1642, 2019. https://doi.org/10.1007/s11263-019-01168-2 147

[21] M. Nagel, R. Amjad, M. Baalen, C. Louizos, and T. Blankevoort, "Up or Down ? Adaptive Rounding for Post-Training Quantization," 2020. https://arxiv.org/pdf/2004.10568.pdf 147

ing.cienc., vol. 16, no. 32, pp. 135–149, julio-diciembre. 2020.

149|