IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 21 November 2023, accepted 8 January 2024, date of publication 10 January 2024, date of current version 22 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3352266

METHODS

Design Space Exploration for Edge Machine Learning Featured by MathWorks FPGA DL Processor: A Survey

STEFANO BERTAZZONI, LORENZO CANESE, GIAN CARLO CARDARILLI, (Life Member, IEEE), LUCA DI NUNZIO, (Member, IEEE), ROCCO FAZZOLARI, MARCO RE, (Member, IEEE), AND SERGIO SPANÒ
Department of Electronic Engineering, Tor Vergata University of Rome, 00133 Rome, Italy

Corresponding author: Sergio Spanò (spano@ing.uniroma2.it)

This work has been supported by the Spoke 1 FutureHPC & BigData of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by Ministero dell'Università e della Ricerca (MUR) Mission 4 - Next Generation EU.

ABSTRACT This paper proposes a Design Space Exploration for Edge machine learning through the utilization of the novel MathWorks FPGA Deep Learning Processor IP, featured in the HDL Deep Learning toolbox. With the ever-increasing demand for real-time machine learning applications, there is a critical need for efficient and low-latency hardware solutions that can operate at the edge of the network, in close proximity to the data source. The HDL Deep Learning toolbox provides a flexible and customizable platform for deploying deep learning models on FPGAs, enabling effective inference acceleration for embedded IoT applications. In this study, our primary focus lies in investigating the impact of parallel processing elements on the performance and resource utilization of the FPGA-based processor. By analyzing the trade-offs between accuracy, speed, energy efficiency, and hardware resource utilization, we aim to gain valuable insights into making optimal design choices for FPGA-based implementations. Our evaluation is conducted on the AMD-Xilinx ZC706 development board, which serves as the target device for our experiments. We consider all the compatible Convolutional Neural Networks available within the HDL Deep Learning toolbox to comprehensively assess the performances.

INDEX TERMS Convolutional neural networks, deep learning, design space exploration, edge machine learning, embedded, FPGA, IoT, machine learning.

I. INTRODUCTION

The deployment of Deep Learning networks on FPGA is one of the most trending topics of recent years' literature, especially if it is targeted to Edge Machine Learning applications [1], [2], [3], [4], [5].

Due to the complexity of meeting the different requirements in such scenarios (e.g. Embedded IoT), several researchers highlighted the need to perform an extensive Design Space Exploration (DSE) taking into account both the target device and network to be implemented [6], [7], [8], [9], [10]. Moreover, different works tried to help the designers proposing some kind of automation for DSE analysis on FPGA-oriented deep learning applications [11], [12], [13], [14].

According to Pham et al. [15], the most complex challenge for the acceleration of CNNs on hardware is the Design Space Exploration phase. This concept has been successively extended by Pham to FPGA devices [16] by proposing the graph in Fig. 1.

The time to design the network is the easiest requirement, mainly due to the plethora of off-the-shelf tools to build, train, and validate a CNN. A medium effort is required when the developer has to deal with the limited amount of memory and the relative clock frequencies of FPGAs. The data communication overhead to exchange the data with the

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

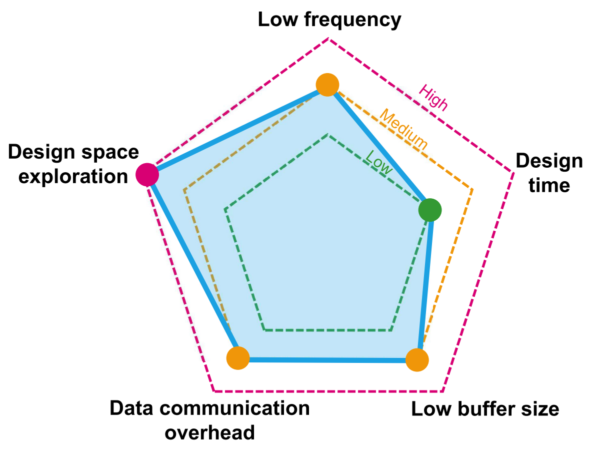


FIGURE 1. Challenges of CNN acceleration on FPGA devices.

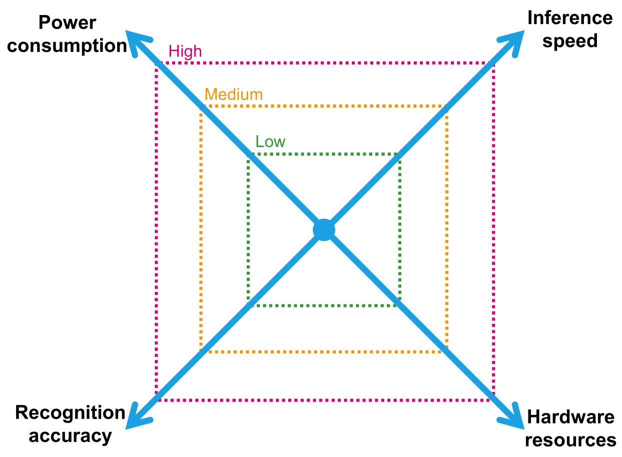


FIGURE 2. Relations between design space exploration parameters.

device is also considered a medium effort challenge. The most complex task is, definitely, DSE which involves the tailoring of all the parameters to match the desired requirements.

DSE parameters for the FPGA implementation of a CNN can be summarized into four and are strictly related to the application constraints. They are: power consumption, availability of hardware resources, inference speed and accuracy. We show in Fig. 2 the relations between them.

For example, if both high inference speed and high recognition accuracy are required, we may need to increase the hardware resources on the device, thus obtaining a high power consumption.

Vice versa, if low power consumption is required, we may allow for a reduction of the recognition accuracy. To keep the power down, we must reduce the hardware resources, thus obtaining a low inference speed.

For the sake of clarity, we oversimplified the concept in Fig. 2 since the relations between the parameters are non-linear and often unpredictable. That is why DSE is an open problem in the field.

The aim of this work is to provide a comprehensive Design Space Exploration targeted to Edge Machine Learning applications. We focus our analysis on the FPGA deployment of the most common CNNs using the novel MATLAB Deep Learning HDL Toolbox [17] on the AMD-Xilinx ZC706 development board. This tool has already been used fruitfully to implement standalone IoT Embedded machine learning systems [18].

A. PAPER ORGANIZATION

This paper is organized as follows.

Section II introduces the MATLAB Deep Learning processor IP and the HDL Deep Learning toolbox.

Section III provides the FPGA implementation results for different configurations of the processor.

Section IV shows the experimental data obtained by the deployment of different CNNs on the target platform.

Finally, sec. V draws the conclusions about the work.

We would highlight that all the data analyzed in this survey can be found in a raw format as attached supplementary material.

II. DEEP LEARNING PROCESSOR

The novel MATLAB Deep Learning HDL Toolbox [17] is a powerful way to deploy Deep Learning applications to FPGA devices via its Deep Learning processor IP [19].

Although the tool is capable of producing hardware code (Verilog and VHDL) that is virtually compatible with every FPGA, the full workflow is available for three development boards [20]: AMD ZCU102, AMD ZC706, and Intel Arria10 SoC. The processor comes in two versions: floating-point single precision and fixed-point 8-bit integer (INT8). The latter requires a quantization process on the networks.

The system can run CNNs, YOLO networks, and LSTMs.

The top-level architecture of the Deep Learning Processor is shown in Fig. 3.

The IP acts as a classic AXI4 slave device. It requires some external DDR which can be accessed via the vendor-dependent memory interface IP. In this case, the processor acts as an AXI4 master. The RAM is needed to store the weights of the neurons, the kernels of the CNN filters, and so on.

The core of the system are its Processing modules:

- **Conv kernel.** It performs Convolution operations. Its performance can be changed by adding or removing physical Threads.
- **FC kernel.** It performs Fully Connected operations. Its performance can be changed by adding or removing physical Threads.
- **Custom kernel.** As the name suggests, it can be programmed to perform the operations of pre-validated custom layers.

The top-level scheduler manages the scheduling of instructions and the retrieval of data from DDR, and determines the appropriate timing to read the data from the RAM. It serves

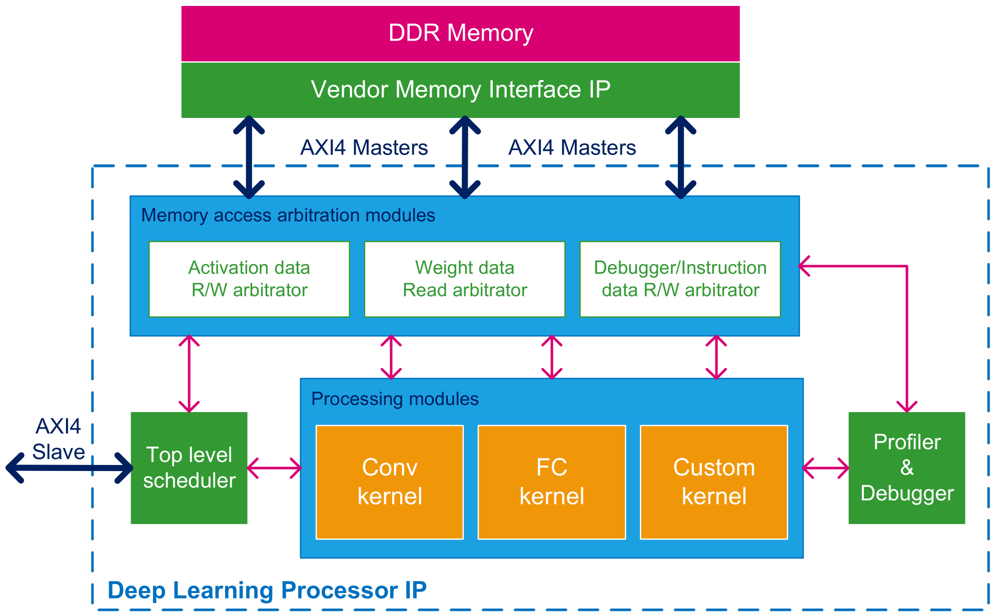

DDR Memory
Vendor Memory Interface IP
AXI4 Masters
AXI4 Masters
Memory access arbitration modules
Activation data R/W arbitrator
Weight data Read arbitrator
Debugger/Instruction data R/W arbitrator
Processing modules
AXI4 Slave
Top level scheduler
Conv kernel
FC kernel
Custom kernel
Profiler & Debugger
Deep Learning Processor IP

FIGURE 3. **Top level architecture of the FPGA Deep Learning Processor IP and its interfaces.**

as the central hub in a distributed computer architecture, responsible for sharing instructions with processing modules.

The Profiler & Debugger module gathers data from the kernels, including the start and stop times of the Conv Kernel, FC Kernel, and other relevant components. Using this data, the profiler module generates a profiler table that summarizes these results.

The Activation and Weight memory access arbitrator modules facilitate the reading and writing of weights and activation data to and from the processing modules. This enables seamless data transfer between these modules. On the other hand, the Profiler relies on its arbitrator to access and manipulate timing data and instructions.

III. FPGA IMPLEMENTATION DATA

As stated in sec. I, we focused our DSE on Edge Machine Learning applications, so we considered the AMD ZC706 development board and the INT8 processor version that allows faster computations without affecting performance [21]. We used Vivado 2020.3 for the synthesis and implementation of the generated VHDL code.

Unfortunately, built-in Toolbox functions are able to give just a rough estimation of required hardware resources. Moreover, no data on Flip-Flops, Look-Up Tables used as RAM, and power dissipation are given. Considering that, our work provides a thorough set of data from real FPGA implementation results.

The default configuration provided by MATLAB R2023a for the chosen platform is as follows.

- System Level Properties
  - TargetPlatform: ‘Xilinx Zynq ZC706 evaluation kit’
  - TargetFrequency: 90
  - SynthesisTool: ‘Xilinx Vivado’
  - ReferenceDesign: ‘AXI-Stream DDR Memory Access : 3-AXIM’
  - SynthesisToolChipFamily: ‘Zynq’
  - SynthesisToolDeviceName: ‘xc7z045’
  - SynthesisToolPackageName: ‘ffg900’
  - SynthesisToolSpeedValue: ‘-2’
- Processor Top Level Properties
  - RunTimeControl: ‘register’
  - RunTimeStatus: ‘register’
  - InputStreamControl: ‘register’
  - OutputStreamControl: ‘register’
  - SetupControl: ‘register’
  - ProcessorDataType: ‘int8’
- Processing Module “conv”
  - ModuleGeneration: ‘on’
  - LRNBlockGeneration: ‘off’
  - SegmentationBlockGeneration: ‘on’
  - ConvThreadNumber: 16
  - InputMemorySize: [227 227 3]
  - OutputMemorySize: [227 227 3]
  - FeatureSizeLimit: 2048
- Processing Module “fc”
  - ModuleGeneration: ‘on’
  - SoftmaxBlockGeneration: ‘off’
  - SigmoidBlockGeneration: ‘off’
  - FCThreadNumber: 8
  - InputMemorySize: 9216
  - OutputMemorySize: 4096
- Processing Module “custom”
  - ModuleGeneration: ‘on’

- Addition: 'on'
- Multiplication: 'on'
- Resize2D: 'off'
- Sigmoid: 'off'
- TanhLayer: 'off'
- InputMemorySize: 40
- OutputMemorySize: 120

As can be seen, the customization capabilities are very high. In our study, we decided to consider the number of Convolutional and Fully Connected threads (ConvThreadNumber and FCThreadNumber) as the design space. This is because they are the most prominent parameters that condition the FPGA implementation results [22].

The available Convolutional threads (from now Conv threads) on the target device are the following "squares": 4, 9, 16, 25, 49, 64. The available Fully Connected threads (from now FC threads) are the following powers of two: 4, 8, 16.

It is very important to note that the IP does not support the combination of 64 Conv and 4 FC threads. For the sake of clarity in the data representation of the following sections, we still show the (64, 4) point as a mere interpolation. The mock data are pointed as a red star in all of the plots to enhance readability.

The data in the following subsections include the processor IP, DDR interfacing, and MathWorks AXI manager IP for external communication [23]. For testing purposes, the board is connected via USB/JTAG to a master PC running MATLAB. The communication speed is up to 30 Mbps, this effectively avoids any communication/computation bottleneck.

A. LOOK UP TABLES

Figure 4 shows the Look Up Tables (LUT) usage.

The LUT required to implement different-sized processors follow a linear trend. This is both considering the increasing number of Conv threads and FC threads. In particular, LUT usage is affected the most by Conv threads.

The minimum LUT usage is about 40% with (4, 4) configuration, while the maximum is about 75% with (64, 16).

B. EMBEDDED RANDOM ACCESS MEMORY

Figures 5 and 6 show the Look Up Tables used as RAM (LUTRAM) and Block RAM (BRAM) usage.

LUTRAM usage tends to have a linear behavior only after 25 Conv threads. This can be easily seen in the FC threads projection plot, where below 25 Conv threads the trend is not inferable.

The minimum LUTRAM usage is about 15% with (4, 8) configuration, while the maximum is about 24% with (64, 16).

BRAMs exhibit a more unusual behavior. While having an almost linear trend if seen from the FC threads projection, the Conv threads relation is very peculiar. 9 and 36 thread

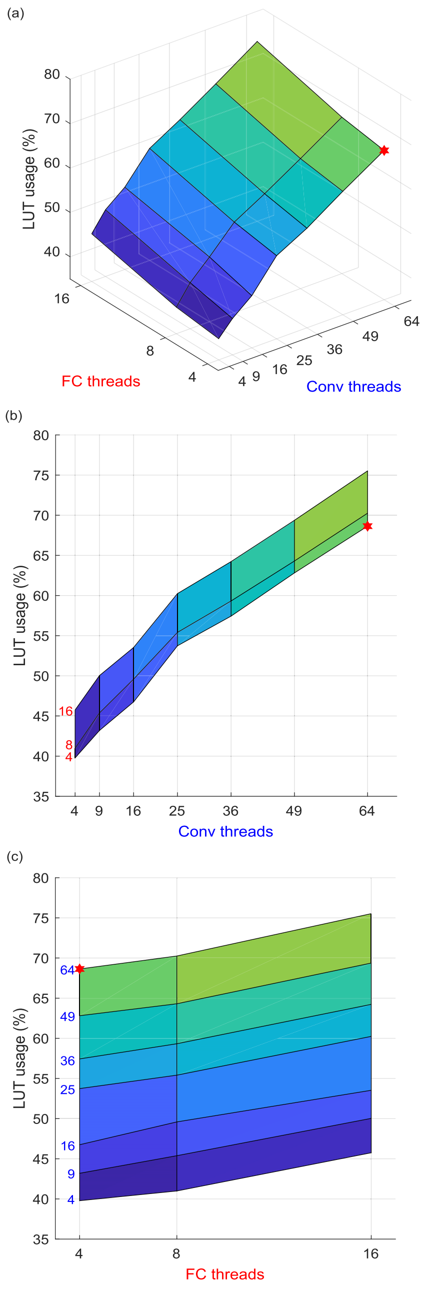


FIGURE 4. Look Up Tables (LUT) utilization for different number of threads. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

versions require the least number of BRAMs, whereas the 25 threads versions are the most demanding. We may speculate that this phenomenon may find its reason considering how the FPGA synthesis deals with mapping the design on fixed-size memory blocks.

The minimum BRAM usage is about 65% with (9, 4) configuration, while the maximum is about 95% with (25, 16).

C. FLIP FLOPS

Figure 7 shows the Flip Flops (FF) usage.

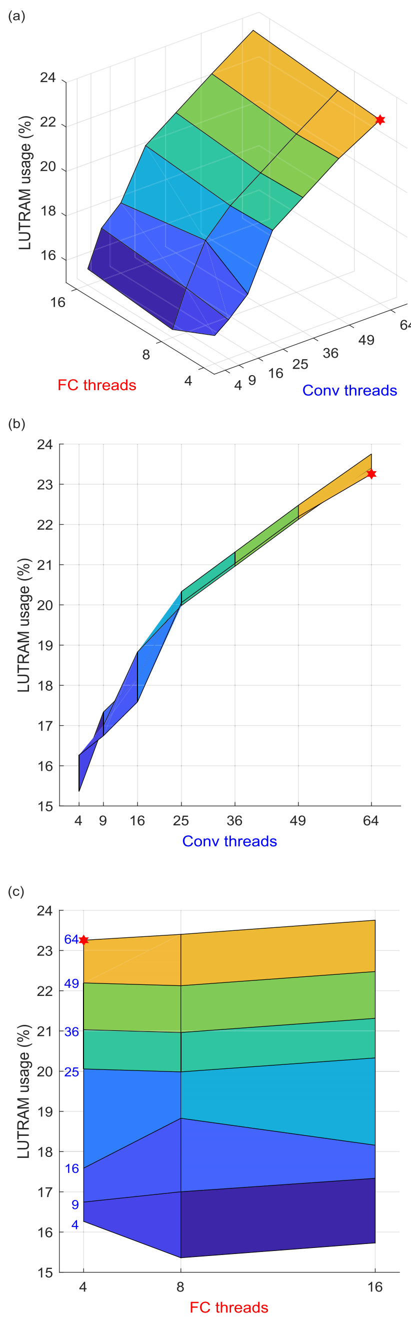


FIGURE 5. Look Up Tables used as RAMs (LUTRAM) utilization for different number of threads. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

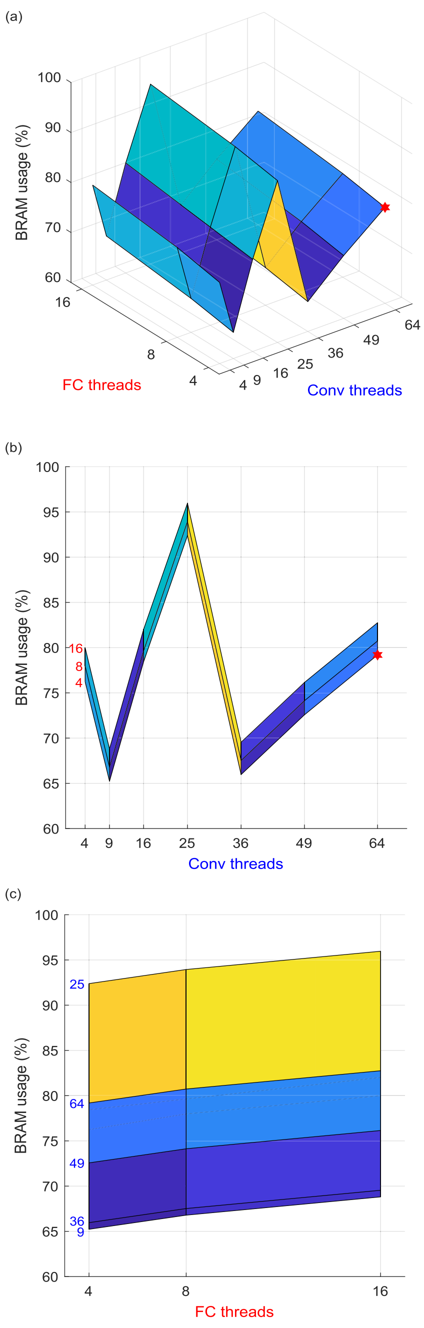


FIGURE 6. Block RAMs (BRAM) utilization for different number of threads. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

FF trends are almost overlapped with those of LUT. There is a usage linear increase with threads, the Conv ones being the most influential with a steeper curve.

The minimum FF usage is about 25% with (4, 4) configuration, while the maximum is about 45% with (64, 16).

D. DIGITAL SIGNAL PROCESSORS

Figure 8 shows the Digital Signal Processor (DSP) usage.

Everything said about LUT and FF applies also to DSP.

The minimum DSP usage is about 15% with (4, 4) configuration, while the maximum is about 90% with (64, 16).

E. POWER CONSUMPTION

Figure 9 shows the dynamic power consumption considering a 90 MHz clock and a vector-less estimation approach. The static power consumption of the FPGA is 268 mW, while the typical consumption of the external RAM is 578 mW.

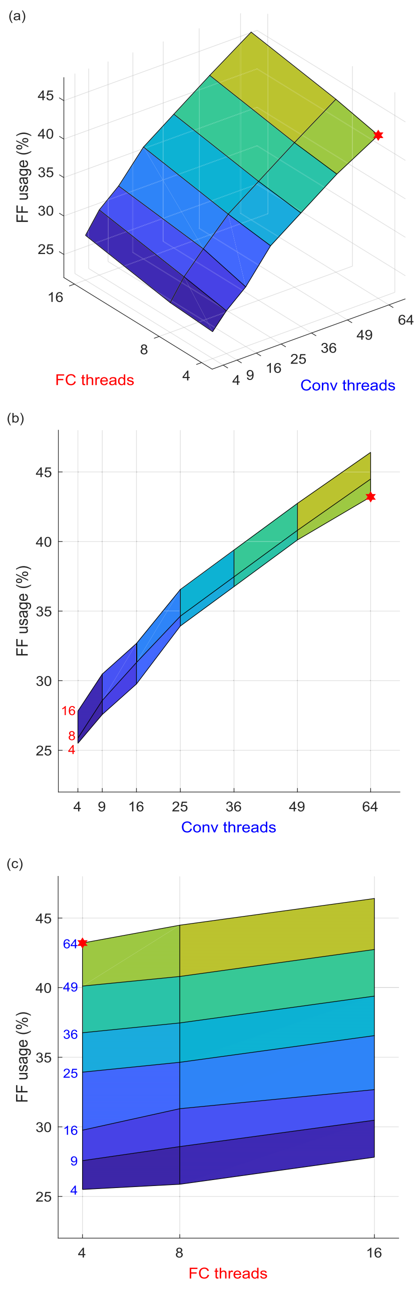


FIGURE 7. Flip Flops (FF) utilization for different number of threads. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

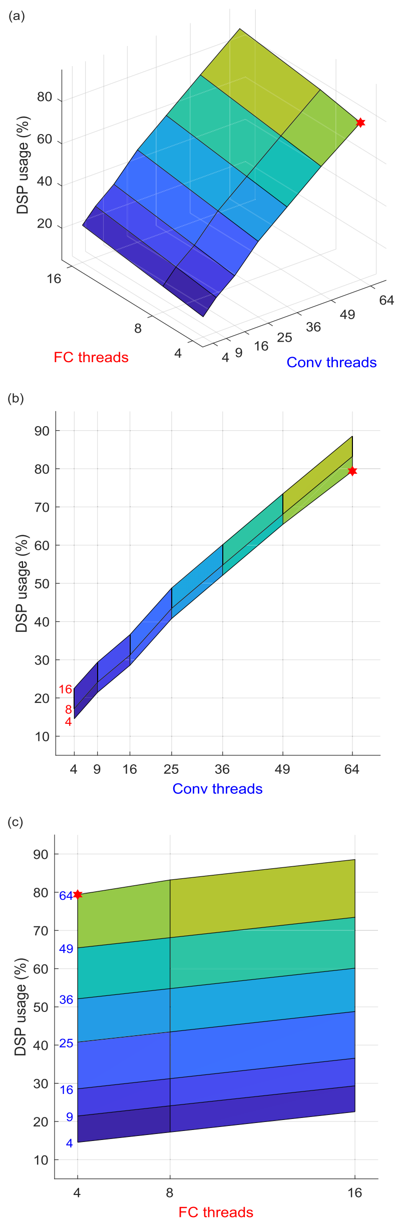


FIGURE 8. Digital Signal Processors (DSP) utilization for different number of threads. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

As expected, power dissipation increases as the number of threads increases. However, we can see a plateau between 25 and 36 Conv threads. Moreover, for 4 FC threads, we see a slight inversion in the trend. Our conjecture is related to BRAM usage, which affects power consumption in such central cases.

The minimum dissipation is about 4.5 W with (4, 4) configuration, while the maximum is about 6.5 W with (64, 16).

In order to assess which hardware resource is the most prominent in contributing to the dynamic power dissipation of the system, we analyzed every implementation. Taking into account a specific resource, we observed how its contribution to the total dissipation is almost independent to the number of Threads of the IP. For this reason, it is possible to average the values among all the considered implementations. In Fig. 10, we show the power percentage for every employed hardware resource.

F. MAXIMUM CLOCK FREQUENCY

In order to asses the hardware resource that limits the performance of the implementations, we analyzed the critical paths that set the maximum achievable clock frequency.

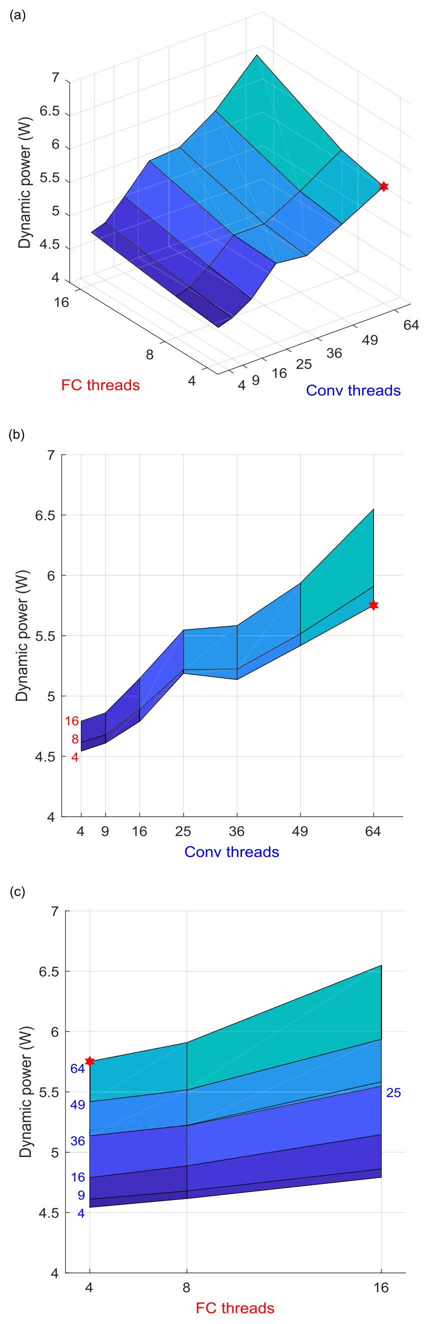


FIGURE 9. Dynamic power consumpion for different number of threads. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

In every case, the critical path resides inside the Convolutional cores and, in particular, the actual hardware resource limiting the maximum clock frequency is the LUT. For this reason, a speed-up of the processing performance may be achieved only by updating the FPGA device since a modification of the internal architecture of the Core is not possible by the Toolbox.

In Table 1 we show how the maximum achievable clock frequency is almost the same for any implementation and it does not depend on the number of Threads.

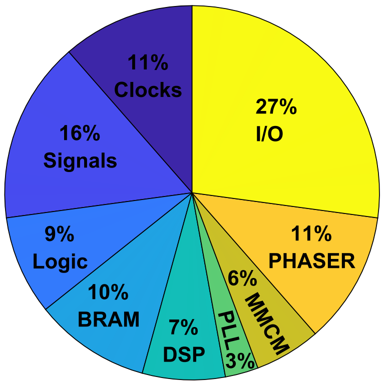


FIGURE 10. Dynamic power consumpion breakdown for every hardware resource.

TABLE 1. Maximum clock frequency achievable by the system for every Conv threads and FC threads combination.

Conv \ FC	4	8	16
4	97 MHz	93 MHz	96 MHz
9	97 MHz	94 MHz	96 MHz
16	97 MHz	94 MHz	96 MHz
25	93 MHz	92 MHz	95 MHz
36	94 MHz	93 MHz	94 MHz
49	94 MHz	91 MHz	92 MHz
64	N/A	91 MHz	92 MHz

G. CONSIDERATIONS ON THE IMPLEMENTATION

According to the previous results, the usage of LUT, FF, and DSP can easily be inferred since their behavior has a linear trend, while the BRAM and power data are quite complex to evaluate.

It is possible to state that Conv threads are the leading parameters when considering the maximum constraints of resources and power of the designer. As a matter of fact, if some FPGA area is required for other parts of an overall system, it is highly suggested that particular attention be paid to Conv threads. Less attention can be paid to the tuning of FC threads.

The presented results must be integrated with the data in the following sections to fully understand how the Design Space affects the CNN performance.

IV. EXPERIMENTS WITH CONVOLUTIONAL NEURAL NETWORKS

In order to assess the performance of the processor, we consider all CNNs for classification tasks available in the toolbox [20]. Only a set of them can actually be deployed with the chosen configuration parameters. We show in Table 2 the successfully tested networks and their limitations in the deployment to FPGA.

For every CNN, in Table 3 we report the accuracy obtained for the ImageNet dataset [24]. These data will be used to give the designer some useful information about the performance of the networks.

TABLE 2. CNNs compatible with the target FPGA device and their limitations.

Network name	Notes
Inception-v3	9 and 36 Conv threads not available
DenseNet-201	No limitations
MobileNet-v2	No limitations
ResNet-18	No limitations
ResNet-50	9 and 36 Conv threads not available
ResNet-101	9 and 36 Conv threads not available
DarkNet-19	No limitations
DarkNet-53	No limitations
SqueezeNet	Only 25 Conv threads available

TABLE 3. CNNs accuracy on ImageNet dataset.

Network name	Accuracy
Inception-v3	77.13%
DenseNet-201	76.32%
MobileNet-v2	70.44%
ResNet-18	69.15%
ResNet-50	74.53%
ResNet-101	75.89%
DarkNet-19	74.04%
DarkNet-53	76.51%
SqueezeNet	55.14%

TABLE 4. Computational overhead of SqueezeNet layers in (25, 8) configuration.

Layer name	Type	Overhead
conv1	2-D Convolution	3.98%
pool1	2-D Max Pooling	3.13%
fire2-squeeze1x1	2-D Convolution	2.15%
fire2-expand1x1	2-D Convolution	2.05%
fire2-expand3x3	2-D Convolution	2.05%
fire3-squeeze1x1	2-D Convolution	5.02%
fire3-expand1x1	2-D Convolution	2.05%
fire3-expand3x3	2-D Convolution	2.05%
pool3	2-D Max Pooling	2.35%
fire4-squeeze1x1	2-D Convolution	1.86%
fire4-expand1x1	2-D Convolution	1.72%
fire4-expand3x3	2-D Convolution	1.72%
fire5-squeeze1x1	2-D Convolution	3.66%
fire5-expand1x1	2-D Convolution	1.72%
fire5-expand3x3	2-D Convolution	1.72%
pool5	2-D Max Pooling	1.23%
fire6-squeeze1x1	2-D Convolution	1.44%
fire6-expand1x1	2-D Convolution	0.96%
fire6-expand3x3	2-D Convolution	0.96%
fire7-squeeze1x1	2-D Convolution	2.12%
fire7-expand1x1	2-D Convolution	0.96%
fire7-expand3x3	2-D Convolution	0.96%
fire8-squeeze1x1	2-D Convolution	2.56%
fire8-expand1x1	2-D Convolution	1.57%
fire8-expand3x3	2-D Convolution	1.57%
fire9-squeeze1x1	2-D Convolution	3.41%
fire9-expand1x1	2-D Convolution	1.57%
fire9-expand3x3	2-D Convolution	1.57%
conv10	2-D Convolution	39.90%
pool10	2-D Global Average Pooling	1.96%

In the following subsections, we provide an evaluation of the CNNs inference speed and their energy efficiency.

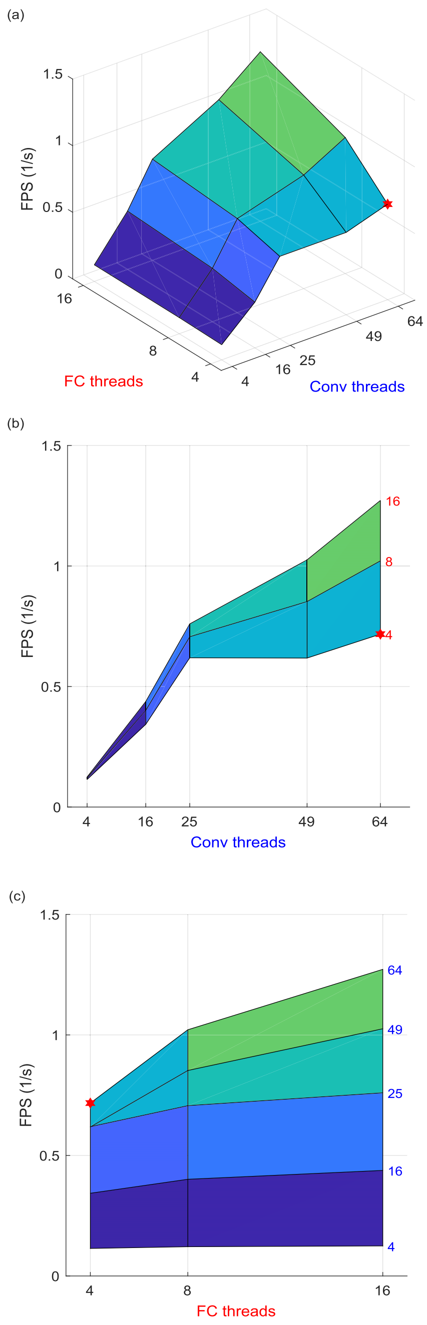


FIGURE 11. Inception-v3 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

Again, it is very important to note that the IP does not support the combination of 64 Conv and 4 FC threads. For the sake of clarity in the data representation of the following sections, we still show the (64, 4) point as a mere interpolation. The mock data are pointed as a red star in all of the plots to enhance readability.

A. INFERENCE SPEED EVALUATION

We evaluated how much time every processor configuration needs to process a frame. The data are given in terms of Frames per Second (FPS) for every CNN considered.

TABLE 5. Computational overhead of Inception-v3 layers in (25, 8) configuration.

Layer name	Type	Overhead
...	...	...
conv2d_2	2-D Convolution	1.08%
conv2d_3	2-D Convolution	1.97%
...	...	...
conv2d_4	2-D Convolution	1.06%
conv2d_5	2-D Convolution	2.99%
...	...	...
conv2d_27	2-D Convolution	2.15%
conv2d_35	2-D Convolution	1.04%
...	...	...
conv2d_32	2-D Convolution	1.04%
...	...	...
conv2d_40	2-D Convolution	1.51%
conv2d_31	2-D Convolution	1.51%
conv2d_45	2-D Convolution	1.26%
...	...	...
conv2d_42	2-D Convolution	1.26%
...	...	...
conv2d_50	2-D Convolution	1.51%
conv2d_41	2-D Convolution	1.51%
conv2d_55	2-D Convolution	1.26%
...	...	...
conv2d_52	2-D Convolution	1.26%
...	...	...
conv2d_60	2-D Convolution	1.51%
conv2d_51	2-D Convolution	1.51%
conv2d_65	2-D Convolution	1.51%
conv2d_66	2-D Convolution	1.09%
conv2d_67	2-D Convolution	1.09%
conv2d_68	2-D Convolution	1.09%
conv2d_69	2-D Convolution	1.09%
conv2d_62	2-D Convolution	1.51%
conv2d_63	2-D Convolution	1.09%
conv2d_64	2-D Convolution	1.09%
...	...	...
conv2d_70	2-D Convolution	1.51%
conv2d_61	2-D Convolution	1.51%
...	...	...
conv2d_73	2-D Convolution	1.51%
conv2d_74	2-D Convolution	1.09%
conv2d_75	2-D Convolution	1.09%
...	...	...
conv2d_71	2-D Convolution	1.51%
...	...	...
conv2d_81	2-D Convolution	2.32%
...	...	...
conv2d_78	2-D Convolution	2.00%
...	...	...
conv2d_85	2-D Convolution	1.05%
conv2d_77	2-D Convolution	1.67%
...	...	...
conv2d_90	2-D Convolution	3.71%
...	...	...
conv2d_87	2-D Convolution	3.19%
...	...	...
conv2d_94	2-D Convolution	1.68%
conv2d_86	2-D Convolution	2.68%
...	...	...

1) SQUEEZENET

Since SqueezeNet runs only on 25 Conv threads, we do not show any plots for it. (25, 4), (25, 8), and (25, 16) configurations for the aforementioned network have the same performance of 8.35 FPS. The SqueezeNet behavior is very peculiar since it is the only network which performance is completely independent of the number of FC threads.

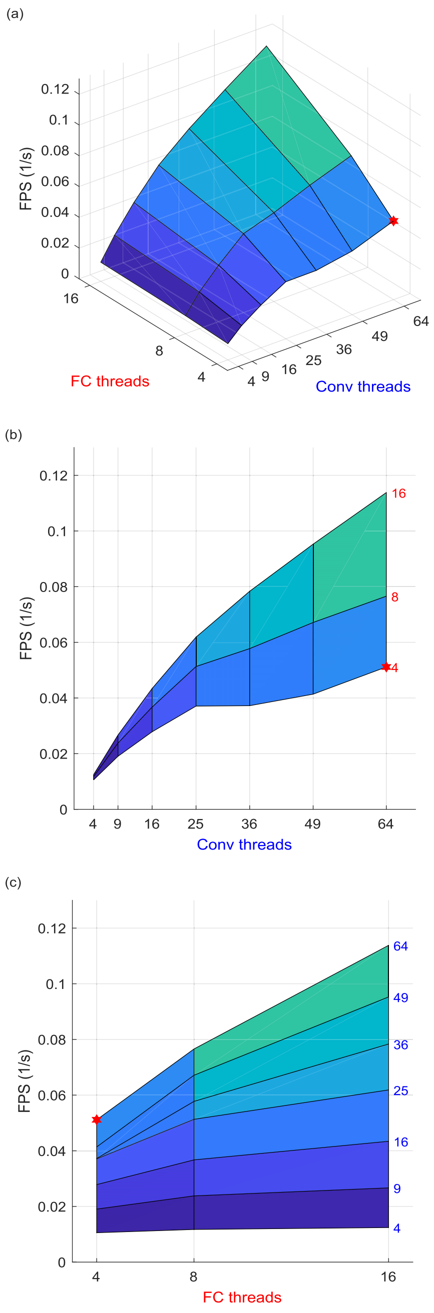


FIGURE 12. DenseNet-201 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

As most representative, we show in Table 4 how much time the processor needs to process each layer of the network in the (25, 8) configuration. The results are normalized considering 100% as the time required to process one frame.

As can be observed, the layer requiring the most time to be processed is the last Convolutional one.

2) INCEPTION-V3

Figure 11 shows the Inception-v3 inference speed.

TABLE 6. Computational overhead of DenseNet-201 layers in (25, 8) configuration.

Layer name	Type	Overhead
...	...	...
conv4_block30_0_bn	Batch Normalization	1.69%
...	...	...
conv4_block31_0_bn	Batch Normalization	1.79%
...	...	...
conv4_block32_0_bn	Batch Normalization	1.88%
...	...	...
conv4_block33_0_bn	Batch Normalization	1.97%
...	...	...
conv4_block34_0_bn	Batch Normalization	2.07%
...	...	...
conv4_block35_0_bn	Batch Normalization	2.17%
...	...	...
conv4_block36_0_bn	Batch Normalization	2.28%
...	...	...
conv4_block37_0_bn	Batch Normalization	2.38%
...	...	...
conv4_block38_0_bn	Batch Normalization	2.48%
...	...	...
conv4_block39_0_bn	Batch Normalization	2.61%
...	...	...
conv4_block40_0_bn	Batch Normalization	2.71%
...	...	...
conv4_block41_0_bn	Batch Normalization	2.84%
...	...	...
conv4_block42_0_bn	Batch Normalization	2.95%
...	...	...
conv4_block43_0_bn	Batch Normalization	3.06%
...	...	...
conv4_block44_0_bn	Batch Normalization	3.20%
...	...	...
conv4_block45_0_bn	Batch Normalization	3.31%
...	...	...
conv4_block46_0_bn	Batch Normalization	3.45%
...	...	...
conv4_block47_0_bn	Batch Normalization	3.58%
...	...	...
conv4_block48_0_bn	Batch Normalization	3.70%
...	...	...
pool4_bn	Batch Normalization	3.85%
...	...	...
pool4_conv	2-D Convolution	1.94%
...	...	...
conv5_block32_0_bn	Batch Normalization	1.02%
...	...	...
bn	Batch Normalization	1.05%
...	...	...

We can see how the FPS growth slows-down when crossing the 25-49 Conf threads line. Moreover, increasing the number of FC threads does not lead to any notable improvement unless more than 49 Conv threads are used.

The minimum speed is about 0.1 FPS with (4, 4) configuration, while the maximum speed is about 1.25 FPS with (64, 16).

As most representative, we show in Table 5 how much time the processor needs to process each layer of the network in the (25, 8) configuration. The results are normalized considering 100% as the time required to process one frame. Please note that only the most demanding layers have been included in the Table.

As can be observed, there is not a promiment layer but, in general, Convolutional ones are the most demanding.

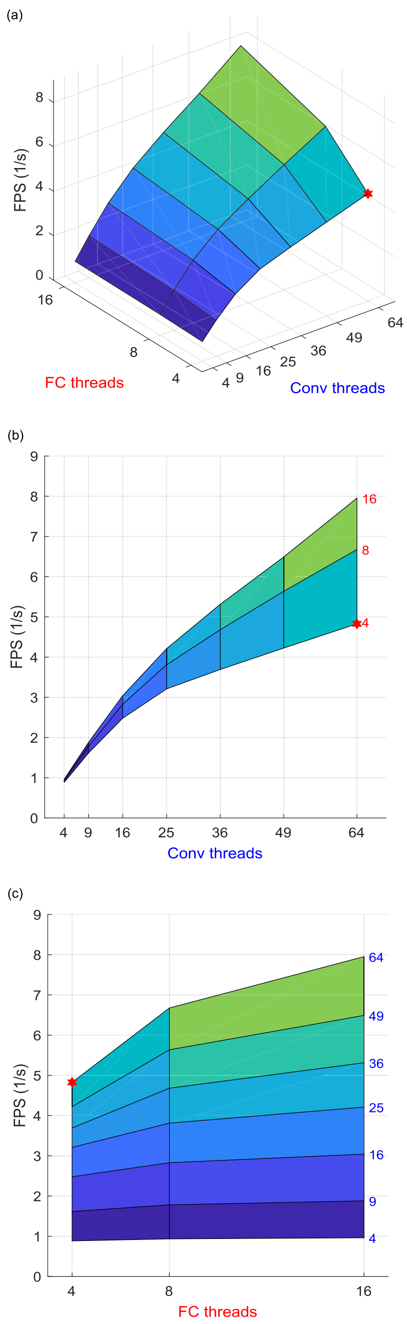


FIGURE 13. MobileNet-v2 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

3) DENSENET-201

Figure 12 shows the DenseNet-201 inference speed.

With this network, the slow-down is prominent only between (25, 4) and (36, 4) threads. In all the other cases, the performance growth is directly proportional to the Conv resources assigned to the processor. In general, the higher the Conv threads are, the more influential the FC threads are; e.g. the (16, 4) configuration is almost the same as (4, 4) and (4, 8) in terms of FPS.

TABLE 7. Computational overhead of MobileNet-v2 layers in (25, 8) configuration.

Layer name	Type	Overhead
Conv1	2-D Convolution	1.13%
expanded_conv_depthwise	2-D Grouped Conv	1.26%
expanded_conv_project	2-D Convolution	2.21%
block_1_expand	2-D Convolution	5.89%
block_1_depthwise	2-D Grouped Conv	2.72%
block_1_project	2-D Convolution	2.03%
block_2_expand	2-D Convolution	2.64%
block_2_depthwise	2-D Grouped Conv	1.84%
block_2_project	2-D Convolution	2.96%
block_2_add	Addition	0.28%
block_3_expand	2-D Convolution	2.64%
block_3_depthwise	2-D Grouped Conv	1.21%
block_3_project	2-D Convolution	0.95%
block_4_expand	2-D Convolution	1.15%
block_4_depthwise	2-D Grouped Conv	0.63%
block_4_project	2-D Convolution	1.26%
block_4_add	Addition	0.10%
block_5_expand	2-D Convolution	1.15%
block_5_depthwise	2-D Grouped Conv	0.63%
block_5_project	2-D Convolution	1.26%
block_5_add	Addition	0.10%
block_6_expand	2-D Convolution	1.15%
block_6_depthwise	2-D Grouped Conv	0.42%
block_6_project	2-D Convolution	0.60%
block_7_expand	2-D Convolution	1.04%
block_7_depthwise	2-D Grouped Conv	0.50%
block_7_project	2-D Convolution	1.17%
block_7_add	Addition	0.05%
block_8_expand	2-D Convolution	1.04%
block_8_depthwise	2-D Grouped Conv	0.50%
block_8_project	2-D Convolution	1.17%
block_8_add	Addition	0.05%
block_9_expand	2-D Convolution	1.04%
block_9_depthwise	2-D Grouped Conv	0.50%
block_9_project	2-D Convolution	1.17%
block_9_add	Addition	0.05%
block_10_expand	2-D Convolution	1.04%
block_10_depthwise	2-D Grouped Conv	0.50%
block_10_project	2-D Convolution	1.64%
block_11_expand	2-D Convolution	2.25%
block_11_depthwise	2-D Grouped Conv	0.76%
block_11_project	2-D Convolution	2.45%
block_11_add	Addition	0.07%
block_12_expand	2-D Convolution	2.25%
block_12_depthwise	2-D Grouped Conv	0.76%
block_12_project	2-D Convolution	2.45%
block_12_add	Addition	0.07%
block_13_expand	2-D Convolution	2.25%
block_13_depthwise	2-D Grouped Conv	0.60%
block_13_project	2-D Convolution	2.13%
block_14_expand	2-D Convolution	3.41%
block_14_depthwise	2-D Grouped Conv	0.53%
block_14_project	2-D Convolution	3.51%
block_14_add	Addition	0.03%
block_15_expand	2-D Convolution	3.41%
block_15_depthwise	2-D Grouped Conv	0.53%
block_15_project	2-D Convolution	3.51%
block_15_add	Addition	0.03%
block_16_expand	2-D Convolution	3.41%
block_16_depthwise	2-D Grouped Conv	0.53%
block_16_project	2-D Convolution	6.74%
Conv_1	2-D Convolution	8.85%
global_average_pooling2d_1	2-D Global Avg Pool	0.45%
Logits	Fully Connected	1.35%

The minimum speed is about 0.15 FPS with (4, 4) configuration, while the maximum speed is about 1.115 FPS with (64, 16). This makes DenseNet-201 the slowest CNN compatible with the toolbox, but also one of the most accurate.

As most representative, we show in Table 6 how much time the processor needs to process each layer of the network in the

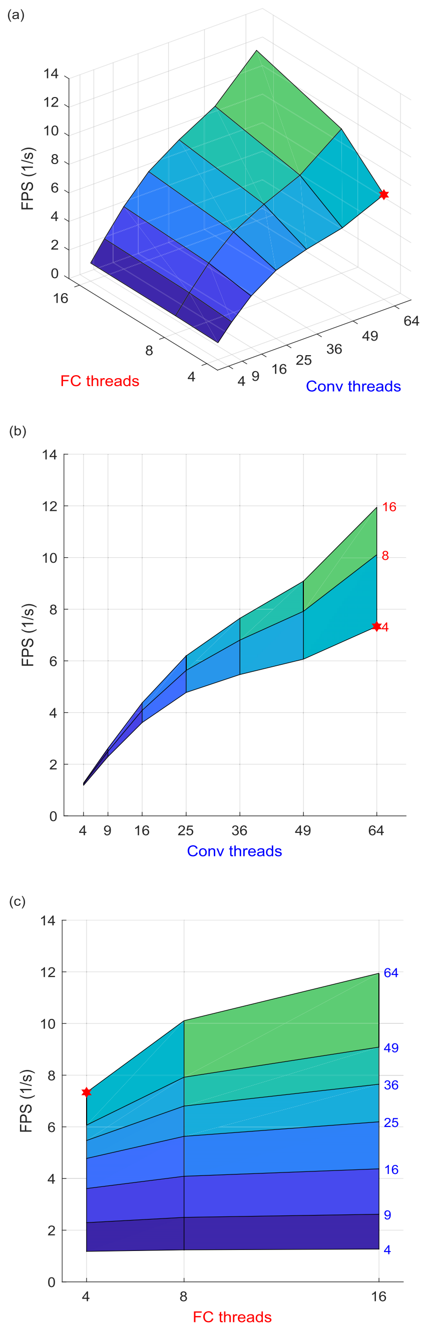


FIGURE 14. ResNet-18 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

(25, 8) configuration. The results are normalized considering 100% as the time required to process one frame. Please note that only the most demanding layers have been included in the Table.

As can be observed, there is not a promiment layer but, in general, Batch Normalization ones are the most demanding.

4) MOBILENET-V2

Figure 13 shows the MobileNet-v2 inference speed.

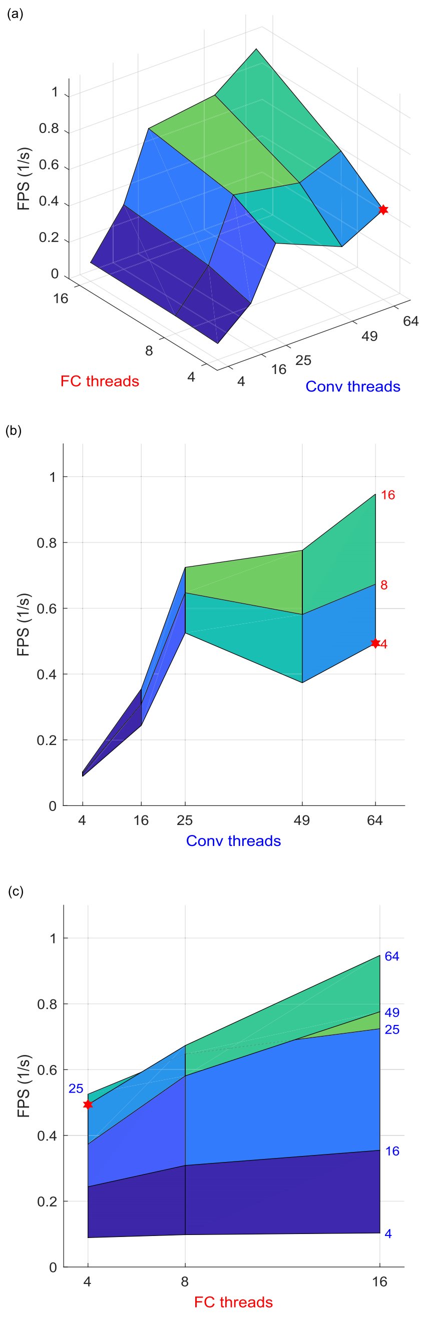


FIGURE 15. ResNet-50 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

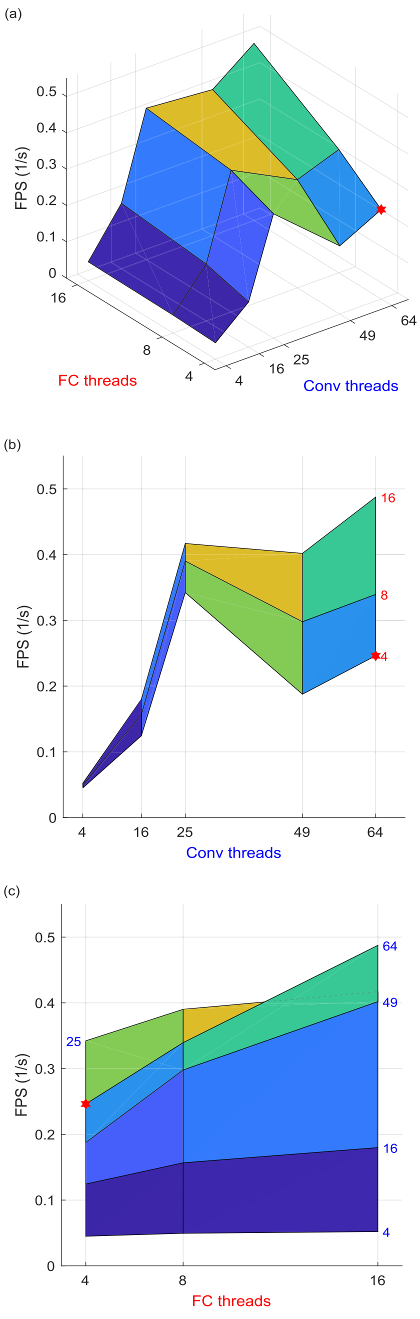


FIGURE 16. ResNet-101 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

The behavior of MobileNet-v2 is the easiest to analyze, since the trend is always growing and it is predictable. Like in the previous network, the higher the Conv threads are, the more influential the FC threads are. A simplification can be deduced; the FC parameter becomes significant from 25 Conv threads and up.

The minimum speed is about 1 FPS with (4, 4) configuration, while the maximum speed is about 8 FPS with (64, 16).

As most representative, we show in Table 7 how much time the processor needs to process each layer of the network in the (25, 8) configuration. The results are normalized considering 100% as the time required to process one frame.

As can be observed, the layer requiring the most time to be processed are the last Convolutional ones.

5) RESNET FAMILY

Figures 14, 15, and 16 show the inference speed of ResNet-18, ResNet-50, and ResNet-101 respectively.

While ResNet-18 has a similar predictable behavior to MobileNet-v2, 50 and 101 versions have a common trend when crossing the 25 to 49 Conv threads line. In fact,

TABLE 8. Computational overhead of ResNet-18 layers in (25, 8) configuration.

Layer name	Type	Overhead
conv1	2-D Convolution	10.83%
pool1	2-D Convolution	2.09%
res2a_branch2a	2-D Convolution	3.94%
res2a_branch2b	2-D Convolution	3.94%
res2a	2-D Convolution	1.07%
res2b_branch2a	2-D Convolution	3.94%
res2b_branch2b	2-D Convolution	3.94%
res2b	2-D Convolution	1.07%
res3a_branch1	2-D Convolution	2.20%
res3a_branch2a	2-D Convolution	2.21%
res3a_branch2b	2-D Convolution	3.83%
res3a	2-D Convolution	0.54%
res3b_branch2a	2-D Convolution	3.83%
res3b_branch2b	2-D Convolution	3.83%
res3b	2-D Convolution	0.54%
res4a_branch1	2-D Convolution	2.10%
res4a_branch2a	2-D Convolution	2.15%
res4a_branch2b	2-D Convolution	3.82%
res4a	2-D Convolution	0.27%
res4b_branch2a	2-D Convolution	3.82%
res4b_branch2b	2-D Convolution	3.82%
res4b	2-D Convolution	0.27%
res5a_branch1	2-D Convolution	4.53%
res5a_branch2a	2-D Convolution	4.55%
res5a_branch2b	2-D Convolution	8.53%
res5a	2-D Convolution	0.13%
res5b_branch2a	2-D Convolution	8.53%
res5b_branch2b	2-D Convolution	8.53%
res5b	2-D Convolution	0.13%
pool5	2-D Global Averaging Pooling	0.27%
fc1000	Fully Connected	0.80%

in almost every case, performance drops-down making the 49 Conv versions slower than the 25 ones. Our conjecture is that, since the BRAM usage is the highest for 25 Conv threads, this leads to an improvement of the inference speed. Therefore, 49 Conv threads are not suggested for both ResNet-50 and ResNet-101.

For ResNet-18, the minimum speed is about 1 FPS with (4, 4) configuration, while the maximum speed is about 14 FPS with (64, 16).

For ResNet-50, the minimum speed is about 0.1 FPS with (4, 4) configuration, while the maximum speed is about 0.6FPS with (64, 16).

For ResNet-101, the minimum speed is about 0.05 FPS with (4, 4) configuration, while the maximum speed is about 0.5 FPS with (64, 16).

As most representatives, we show in Tables 8, 9, and 10 how much time the processor needs to process each layer of the networks in the (25, 8) configuration. The results are normalized considering 100% as the time required to process one frame.

As can be observed, the layer requiring the most time to be processed is the first Convolutional one, followed by some in the bottom third of the stack.

As can be observed, the layers requiring the most time to be processed are some Convolutional ones but, unlike ResNet-18, they are spread throught the stack.

TABLE 9. Computational overhead of ResNet-50 layers in (25, 8) configuration.

Layer name	Type	Overhead
...	...	...
res3a_branch1	2-D Convolution	3.46%
...	...	...
res3a_branch2c	2-D Convolution	1.86%
...	...	...
res3b_branch2a	2-D Convolution	1.93%
...	...	...
res3b_branch2c	2-D Convolution	1.86%
...	...	...
res3c_branch2a	2-D Convolution	1.93%
...	...	...
res3c_branch2c	2-D Convolution	1.86%
...	...	...
res3d_branch2a	2-D Convolution	1.93%
...	...	...
res3d_branch2c	2-D Convolution	1.86%
...	...	...
res4a_branch1	2-D Convolution	7.83%
res4a_branch2a	2-D Convolution	2.14%
...	...	...
res5a_branch1	2-D Convolution	7.63%
res5a_branch2a	2-D Convolution	2.01%
...	...	...
res5a_branch2c	2-D Convolution	3.82%
...	...	...
res5b_branch2a	2-D Convolution	3.86%
...	...	...
res5b_branch2c	2-D Convolution	3.82%
...	...	...
res5c_branch2a	2-D Convolution	3.86%
...	...	...
res5c_branch2c	2-D Convolution	3.82%
...	...	...
res3a_branch1	2-D Convolution	3.46%
...	...	...
res3a_branch2c	2-D Convolution	1.86%
...	...	...
res3b_branch2a	2-D Convolution	1.93%
...	...	...
res3b_branch2c	2-D Convolution	1.86%
...	...	...
res3c_branch2a	2-D Convolution	1.93%
...	...	...
res3c_branch2c	2-D Convolution	1.86%
...	...	...
res3d_branch2a	2-D Convolution	1.93%
...	...	...
res3d_branch2c	2-D Convolution	1.86%
...	...	...
res4a_branch1	2-D Convolution	7.83%
res4a_branch2a	2-D Convolution	2.14%
...	...	...
res5a_branch1	2-D Convolution	7.63%
res5a_branch2a	2-D Convolution	2.01%
...	...	...
res5a_branch2c	2-D Convolution	3.82%
...	...	...
res5b_branch2a	2-D Convolution	3.86%
...	...	...
res5b_branch2c	2-D Convolution	3.82%
...	...	...
res5c_branch2a	2-D Convolution	3.86%
...	...	...
res5c_branch2c	2-D Convolution	3.82%
...	...	...

As can be observed, the layers requiring the most time to be processed are some Convolutional ones but, unlike

TABLE 10. Computational overhead of ResNet-101 layers in (25, 8) configuration.

Layer name	Type	Overhead
...	...	...
res2b_branch2a	2-D Convolution	1.11%
...	...	...
res2c_branch2a	2-D Convolution	1.11%
...	...	...
res3a_branch1	2-D Convolution	2.09%
...	...	...
res3a_branch2c	2-D Convolution	1.12%
...	...	...
res3b1_branch2a	2-D Convolution	1.17%
...	...	...
res3b1_branch2c	2-D Convolution	1.12%
...	...	...
res3b2_branch2a	2-D Convolution	1.17%
...	...	...
res3b2_branch2c	2-D Convolution	1.12%
...	...	...
res3b3_branch2a	2-D Convolution	1.17%
...	...	...
res3b3_branch2c	2-D Convolution	1.12%
...	...	...
res4a_branch1	2-D Convolution	4.72%
res4a_branch2a	2-D Convolution	1.29%
...	...	...
res5a_branch1	2-D Convolution	4.60%
res5a_branch2a	2-D Convolution	1.21%
...	...	...
res5a_branch2c	2-D Convolution	2.31%
...	...	...
res5b_branch2a	2-D Convolution	2.33%
...	...	...
res5b_branch2c	2-D Convolution	2.31%
...	...	...
res5c_branch2a	2-D Convolution	2.33%
...	...	...
res5c_branch2c	2-D Convolution	2.31%
...	...	...

ResNet-18 and ResNet-50, they can be found in the middle of the stack.

6) DARKNET FAMILY

Figures 17 and 18 show the inference speed of DarkNet-19 and DarkNet-53 respectively.

The behavior of both networks is overlappable. The plateau/drop in performance after 25 Conv threads is also present in this family of CNNs. The 36 Conv thread versions are less favorable in terms of inference speed.

For DarkNet-19, the minimum speed is about 0.5 FPS with (4, 4) configuration, while the maximum speed is about 5 FPS with (64, 16).

For DarkNet-53, the minimum speed is 0.2 about FPS with (4, 4) configuration, while the maximum speed is about 2 FPS with (64, 16).

As most representatives, we show in Tables 11 and 12 how much time the processor needs to process each layer of the networks in the (25, 8) configuration. The results are normalized considering 100% as the time required to process one frame.

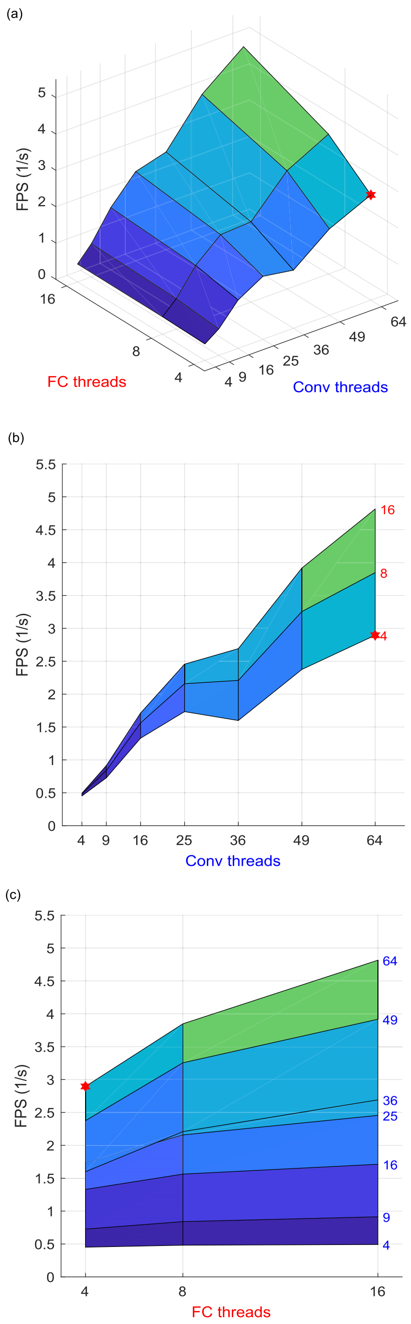


FIGURE 17. DarkNet-19 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

As can be observed, the layer requiring the most time to be processed is the last Convolutional one.

Again, As can be observed, the layer requiring the most time to be processed is the last Convolutional one.

7) PERFORMANCE-ACCURACY ANALYSIS

We evaluated the accuracy of the considered networks with respect to their speed in order to assess the best trade-off for every application. We found that the relations are the same for every thread configuration.

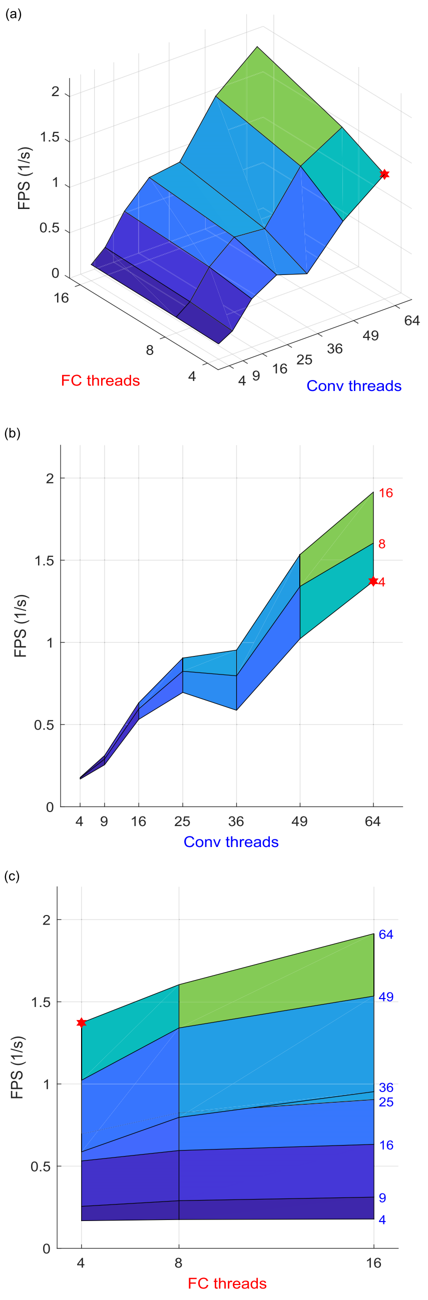


FIGURE 18. DarkNet-53 Frames Per Second (FPS). (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

TABLE 11. Computational overhead of DarkNet-19 layers in (25, 8) configuration.

Layer name	Type	Overhead
conv1	2-D Convolution	2.51%
pool1	2-D Max Pooling	2.03%
conv2	2-D Convolution	4.66%
pool2	2-D Max Pooling	1.20%
conv3	2-D Convolution	3.94%
conv4	2-D Convolution	4.02%
conv5	2-D Convolution	3.94%
pool3	2-D Max Pooling	0.72%
conv6	2-D Convolution	4.25%
conv7	2-D Convolution	4.71%
conv8	2-D Convolution	4.25%
pool4	2-D Max Pooling	0.59%
conv9	2-D Convolution	3.57%
conv10	2-D Convolution	3.62%
conv11	2-D Convolution	3.57%
conv12	2-D Convolution	3.62%
conv13	2-D Convolution	3.57%
pool5	2-D Max Pooling	0.28%
conv14	2-D Convolution	6.44%
conv15	2-D Convolution	6.49%
conv16	2-D Convolution	6.44%
conv17	2-D Convolution	6.49%
conv18	2-D Convolution	6.44%
conv19	2-D Convolution	12.41%
avg1	2-D Global Aberage Pooling	0.22%

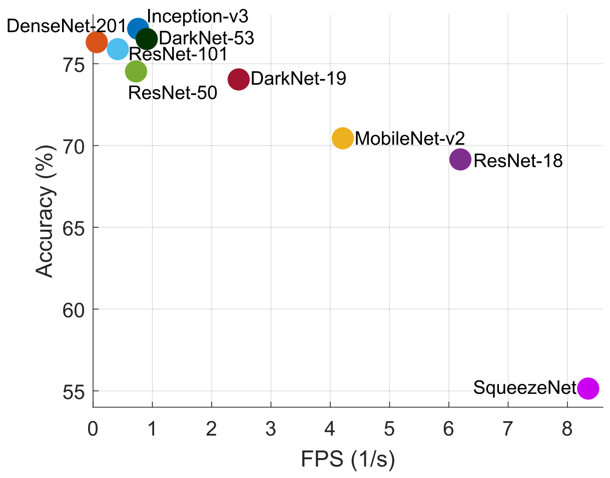


FIGURE 19. Accuracy on ImageNet dataset vs Frame Per Second (FPS) for 25 Convolutional and 16 Fully Connected threads.

For the sake of clarity, in Fig. 19 we show only the (25, 16) processor results, which is compatible with all the CNNs.

SqueezeNet is the best choice if the application requires high speed; however, its accuracy is the lowest one. MobileNet-v2 and DarkNet-19 would be the best trade-offs when both good accuracy and speed are required. If speed is not a critical constraint, but accuracy is, we think Inception-v3 would be the best designer choice.

These results are comparable with other similar works [22].

8) CONSIDERATIONS ON PERFORMANCE

According to the previous results, the speed of the considered network is related to the size of the processor. However, in ResNet and DarkNet families, 36 and 49 Conv threads configurations may show a performance drop. For such reason, it would be better to avoid such parameters and to take into consideration only 64 Conv threads setups when higher speeds are mandatory.

It is possible to state that Conv threads are the leading parameters when considering the inference speed. Again, it is highly suggested that particular attention be paid to Conv threads. Less attention can be paid to the tuning of FC threads, especially when less than 25 Conv threads are used.

TABLE 12. Computational overhead of DarkNet-53 layers in (25, 8) configuration.

Layer name	Type	Overhead
...	...	...
conv2	2-D Convolution	2.18%
conv3	2-D Convolution	1.82%
conv4	2-D Convolution	1.78%
...	...	...
conv5	2-D Convolution	1.89%
conv6	2-D Convolution	1.54%
conv7	2-D Convolution	1.50%
...	...	...
conv8	2-D Convolution	1.54%
conv9	2-D Convolution	1.50%
...	...	...
conv10	2-D Convolution	1.58%
conv11	2-D Convolution	1.80%
conv12	2-D Convolution	1.62%
...	...	...
conv13	2-D Convolution	1.80%
conv14	2-D Convolution	1.62%
...	...	...
conv15	2-D Convolution	1.80%
conv16	2-D Convolution	1.62%
...	...	...
conv17	2-D Convolution	1.80%
conv18	2-D Convolution	1.62%
...	...	...
conv19	2-D Convolution	1.80%
conv20	2-D Convolution	1.62%
...	...	...
conv21	2-D Convolution	1.80%
conv22	2-D Convolution	1.62%
...	...	...
conv23	2-D Convolution	1.80%
conv24	2-D Convolution	1.62%
...	...	...
conv25	2-D Convolution	1.80%
conv26	2-D Convolution	1.62%
...	...	...
conv27	2-D Convolution	3.06%
...	...	...
conv44	2-D Convolution	2.51%
conv45	2-D Convolution	2.48%
conv46	2-D Convolution	2.46%
...	...	...
conv47	2-D Convolution	2.48%
conv48	2-D Convolution	2.46%
...	...	...
conv49	2-D Convolution	2.48%
conv50	2-D Convolution	2.46%
...	...	...
conv51	2-D Convolution	2.48%
conv52	2-D Convolution	2.46%
...	...	...
conv53	2-D Convolution	4.64%

We also highlighted that speed and accuracy are not necessarily related, since the most accurate network can not necessarily be the slowest one (e.g. Inception-v3).

B. ENERGY EFFICIENCY

We evaluated how much energy every processor configuration needs to process a frame. The data are given in terms of energy per frame (J) for every CNN considered.

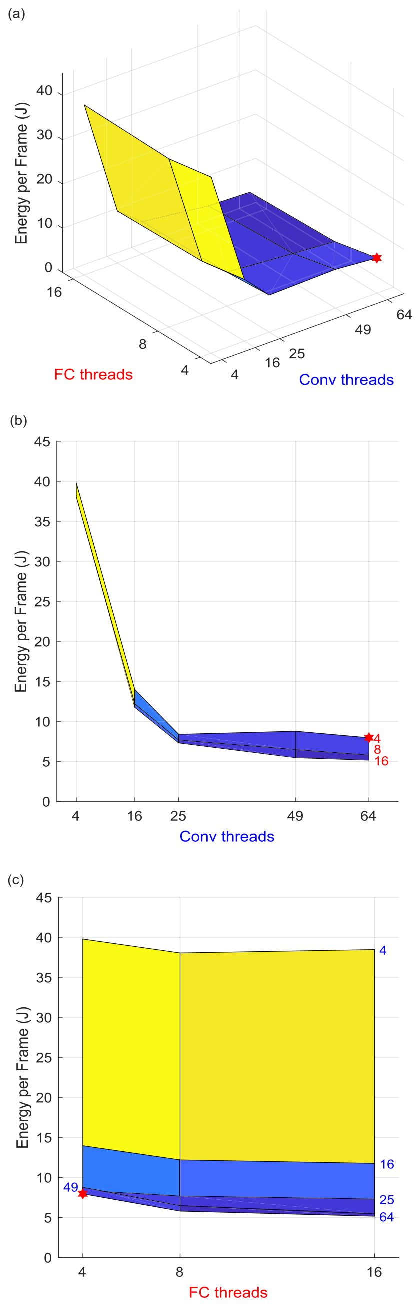


FIGURE 20. Inception-v3 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

1) SQUEEZENET

Since SqueezeNet runs only on 25 Conv threads, we do not show any plots for it. Below are the results for the CNN.

- (25, 4) - 0.621 J
- (25, 8) - 0.625 J
- (25, 16) - 0.664 J

The SqueezeNet behavior is very peculiar since it is the only network which energy increases with the number of FC threads.

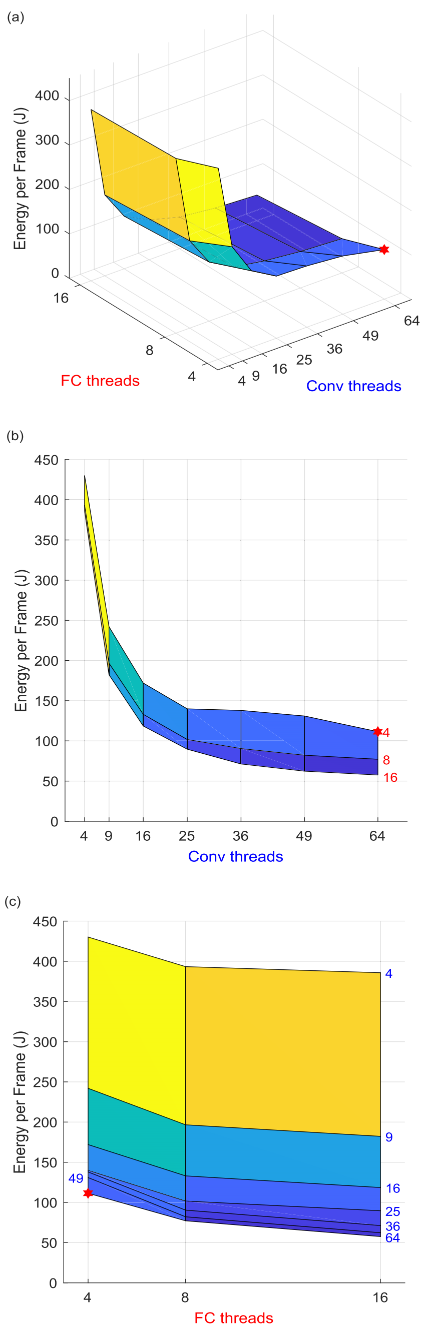


FIGURE 21. DenseNet-201 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

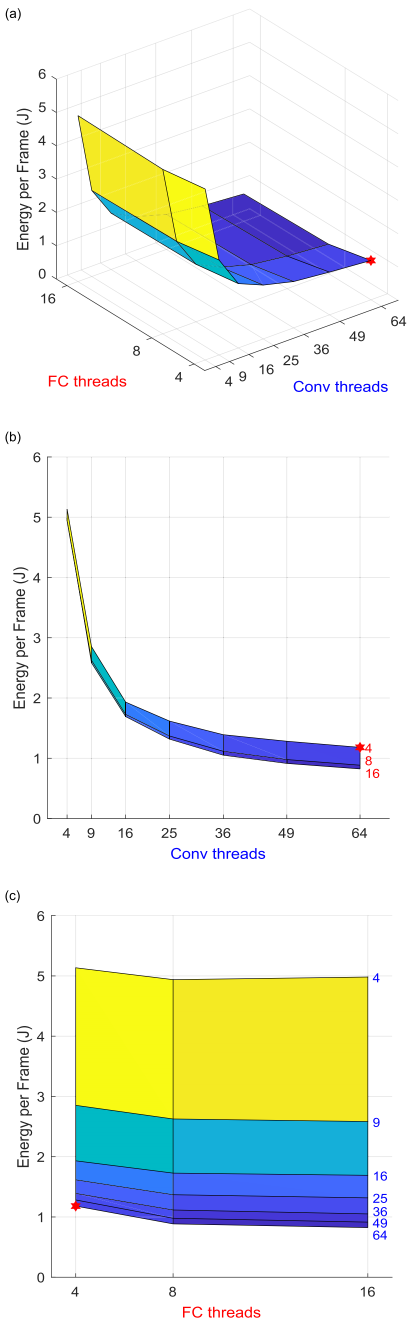


FIGURE 22. MobileNet-v2 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

2) INCEPTION-V3

Figure 20 shows the Inception-v3 energy requirements.

While FC threads are not as influential, we can see a huge energy drop-off from 4 to 25 Conv threads. There is little to not benefit from using a high number of Convs.

The minimum energy to process one frame is about 5 J with (64, 16) configuration, while the maximum energy is about 40 J with (4, 4).

3) DENSENET-201

Figure 21 shows the DenseNet-201 energy requirements.

In this case, FC threads make a substantial difference in energy, especially above 9 Conv threads. Other considerations can be transposed from Inception-v3 network.

The minimum energy to process one frame is about 50 J with (64, 16) configuration, while the maximum energy is about 425 J with (4, 4).

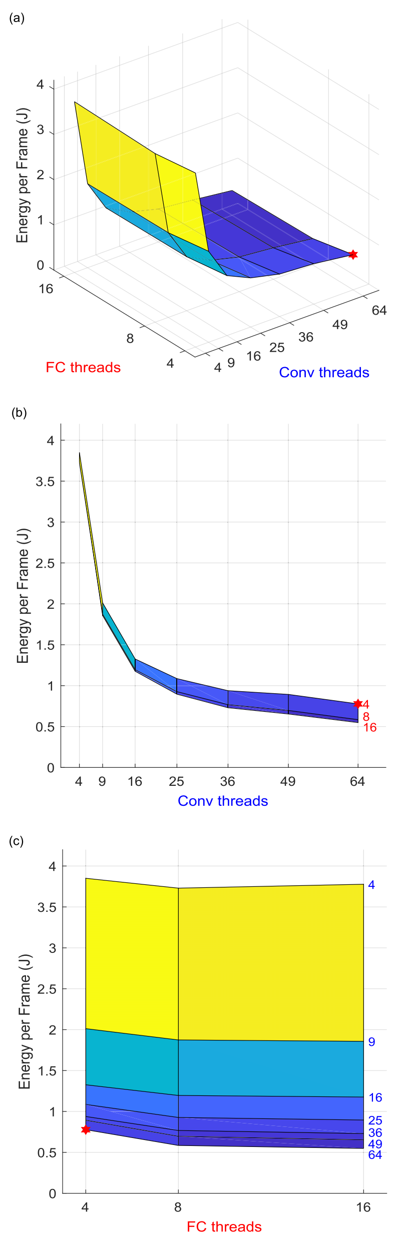


FIGURE 23. ResNet-18 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

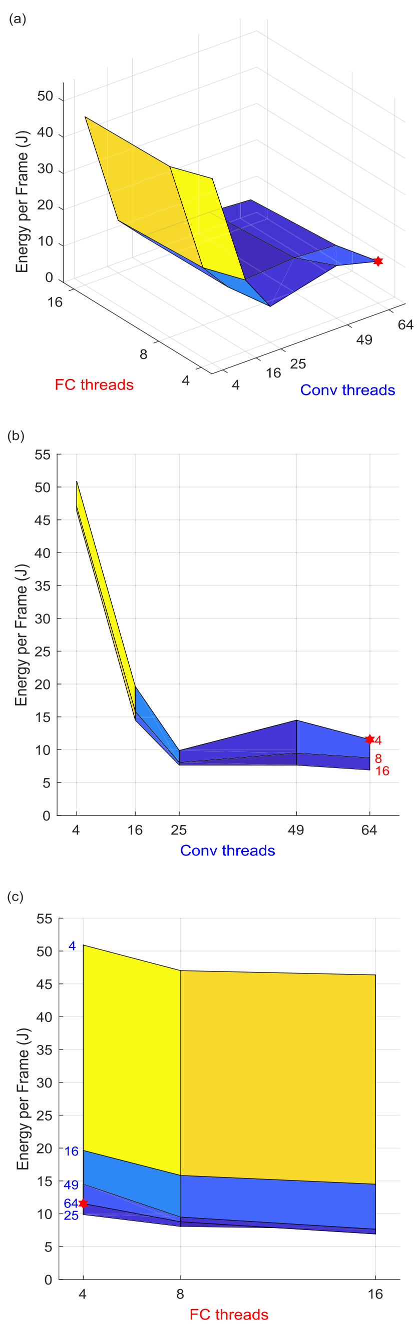


FIGURE 24. ResNet-50 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

4) MOBILENET-V2

Figure 22 shows the MobileNet-v2 energy requirements.

Same considerations apply to this CNN but there is no advantage in using 16 FC threads instead of 8. Furthermore, the (4, 16) configuration requires more energy than the (8, 16) one.

The minimum energy to process one frame is about 1 J with (64, 16) configuration, while the maximum energy is about 5 J with (4, 4).

5) RESNET FAMILY

Figures 23, 24, and 25 show the energy requirements of ResNet-18, ResNet-50, and ResNet-101 respectively.

ResNet-18 considerations are the same of MobileNet-v2 ones, however, the other two CNNs exhibit a similar different behavior. After 25 Conv threads, the energy requirements increase, making 49 and 64 configurations less efficient. Moreover, it is more favorable to upgrade from 4 to 8 FC threads than using 16 versions.

For ResNet-18, the minimum energy to process one frame is about 0.5 J with (64, 16) configuration, while the maximum energy is about 3.75 J with (4, 4). For ResNet-50, the minimum energy to process one frame is about 7.5 J with (64, 16) configuration, while the maximum energy is about 50 J with (4, 4). For ResNet-101, the minimum energy

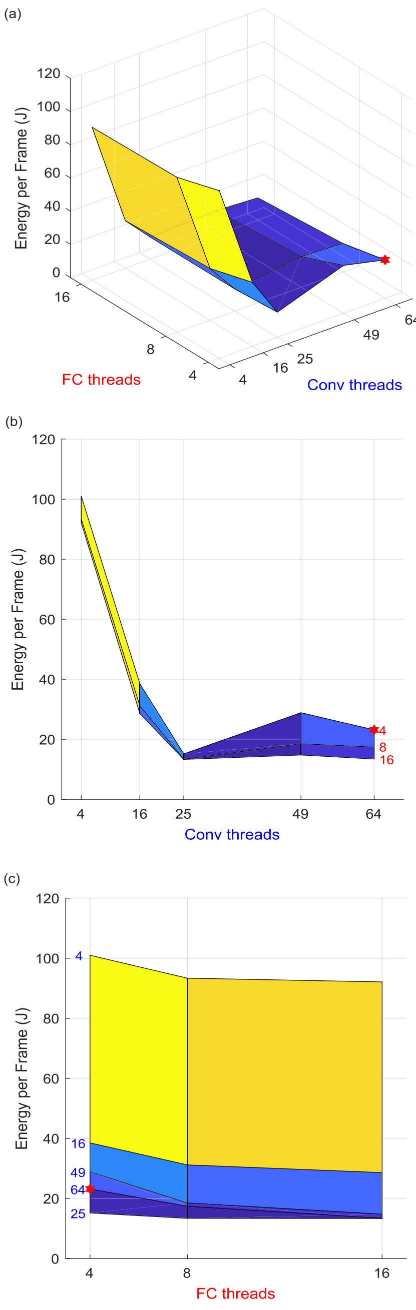


FIGURE 25. ResNet-101 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

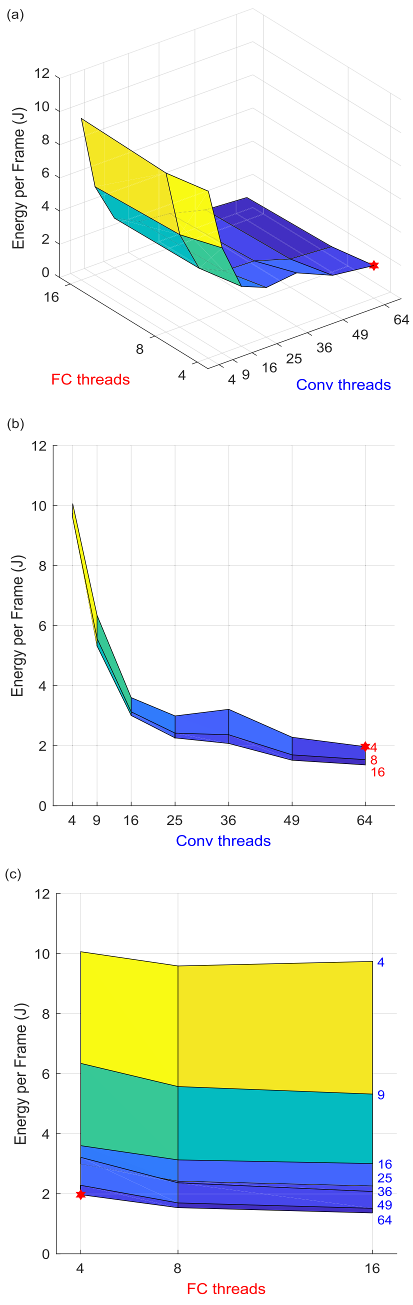


FIGURE 26. DarkNet-19 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.

to process one frame is about 15 J with (64, 16) configuration, while the maximum energy is about 100 J with (4, 4).

6) DARKNET FAMILY

Figures 26 and 27 show the energy requirements of DarkNet-19 and DarkNet-53 respectively.

The trends are almost superimposable, indeed, energy has a decreasing pace for Conv threads with the exception of 36 configurations where the requirements may increase. This makes 36 Conv threads an unfavorable choice for designers. Like the deepest ResNets, it is more favorable to upgrade from 4 to 8 FC threads than using 16 versions.

For DarkNet-19, the minimum energy to process one frame is about 1.5 J with (64, 16) configuration, while the maximum energy is about 10J with (4, 4). For ResNet-50, the minimum energy to process one frame is about 3 J with (64, 16) configuration, while the maximum energy is about 27 J with (4, 4).

7) ENERGY-ACCURACY ANALYSIS

We evaluated the accuracy of the considered networks with respect to their energy requirements in order to assess the best trade-off for every application. We found that the relations are the same for every thread configuration.

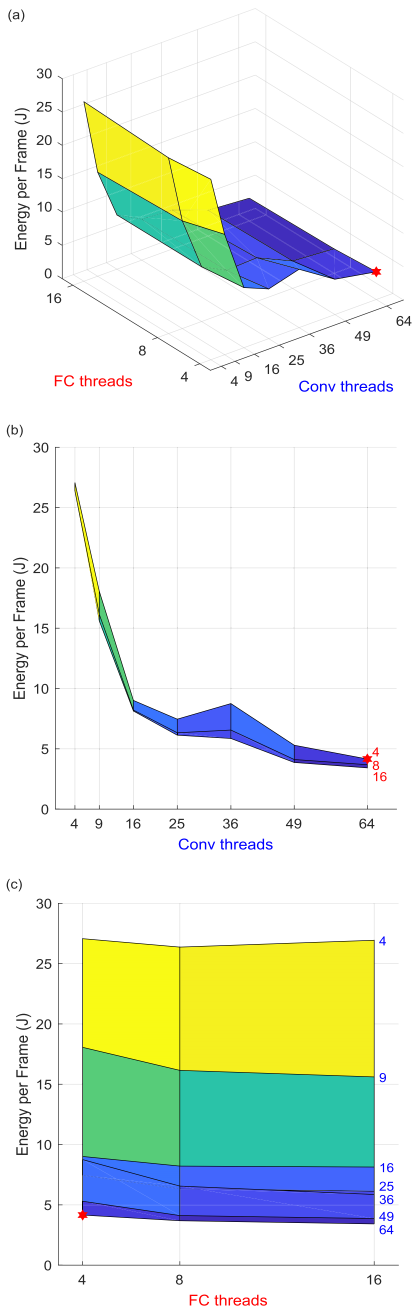


FIGURE 27. **DarkNet-53 energy required to process one frame. (a) 3D surface. (b) Conv threads axis projection. (c) FC threads axis projection.**

For the sake of clarity, in Fig. 28 we show only the (25, 16) processor results which is compatible with all the CNNs.

The plot shows how the relations, if inverted, are the same of Fig. 19. This means that SqueezeNet is the best choice if the application requires low energy, however, its accuracy is the lowest one. MobileNet-v2 and DarkNet-19 would be the best trade-offs when both good accuracy and energy efficiency. If energy is not a critical constraint but accuracy is, we think Inception-v3 would be the best designer choice.

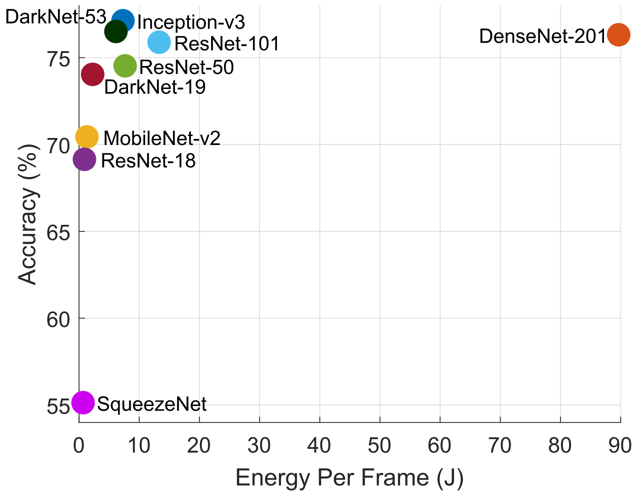


FIGURE 28. **Accuracy on ImageNet dataset vs energy per frame for 25 Convolutional and 16 Fully Connected threads.**

8) CONSIDERATIONS ON ENERGY EFFICIENCY

According to the previous results, the energy requirements of the considered networks are related to the size of the processor. In general, the more threads, the less energy is required to process a frame. However, in ResNet and DarkNet families, 36 and 49 Conv threads configurations may show reduced efficiency. For such reasons, it would be better to avoid such parameters and to take into consideration only 64 Conv thread setups when demanding constraints are mandatory.

It is possible to state that the Conv threads are the leading parameters when considering the energy efficiency. Again, it is highly suggested that particular attention be paid to Conv threads. Less attention can be paid to the tuning of FC threads, especially when more than 25 Conv threads are used.

We also highlighted that energy and accuracy are not necessarily related, since the most accurate network can not necessarily be the most energy-demanding one (e.g. Inception-v3).

V. CONCLUSION

We provided a Design Space Exploration analysis targeted to Edge Machine Learning applications. We focused our work on the FPGA deployment of the most common CNNs using the novel MATLAB Deep Learning HDL Toolbox on the AMD-Xilinx ZC706 development board.

Overall, the Convolutional threads of the implemented Deep Learning Processor IP are the leading parameter to tune the required hardware resources, energy efficiency, and inference speed. The choice of the number of Fully Connected threads does not make a real difference in the general performance. Moreover, we discourage the use of medium-sized systems (i.e. 25, 36, and 49 Conv threads) since their performance may not be improved if compared to small-sized ones.

In order to evaluate the amount of FPGA resources needed by the design, BRAM usage can be considered as the worst case. The only exception is in 64 Conv threads configuration, where the worst usage case depends on DSP. Since power dissipation is always between 4.5W and 7.5W, we can state that the system is suitable for Edge, embedded, and IoT use cases.

Among the compatible CNNs for classification tasks, SqueezeNet is the fastest, while DenseNet-201 is the slowest. It is important to note that, according to our study, faster inference speed does not necessarily imply higher recognition accuracy. Inception-v3 is the best trade-off in terms of speed/accuracy balance.

Concerning energy, we can state that the maximum configuration (64, 16) has the best figure for energy efficiency. In general, bigger systems have better efficiencies. Moreover, we observed that the faster a network is, the lower is the energy required to process one frame.

This paper proposes to be an aid for developers in implementing CNN on FPGA devices using automated CAD tools. We have shown how, depending on the need and application, performance and power consumption inherent to Edge Computing can be achieved [5].

Future developments will include the study of LSTM-type networks, and the development of a software toolbox to find the best combination of parameters from the specifications given by the designer.

ACKNOWLEDGMENT

This work has been supported by the Spoke 1 FutureHPC & BigData of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Mission 4 - Next Generation EU. The authors would like to thank Advanced Micro Devices Inc. (AMD) for providing the FPGA hardware and software tools with the AMD-Xilinx University Program.

REFERENCES

[1] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[2] G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, A. Ricci, and S. Spano, "An FPGA-based multi-agent reinforcement learning timing synchronizer," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107749.

[3] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[4] Z. Xing, S. Zhao, W. Guo, F. Meng, X. Guo, S. Wang, and H. He, "Coal resources under carbon peak: Segmentation of massive laser point clouds for coal mining in underground dusty environments using integrated graph deep learning model," *Energy*, vol. 285, Dec. 2023, Art. no. 128771.

[5] V. H. Kim and K. K. Choi, "A reconfigurable CNN-based accelerator design for fast and energy-efficient object detection system on mobile FPGA," *IEEE Access*, vol. 11, pp. 59438–59445, 2023.

[6] C. Sestito, S. Perri, and R. Stewart, "Design-space exploration of quantized transposed convolutional neural networks for FPGA-based systems-on-chip," in *Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Sep. 2022, pp. 1–6.

[7] L. R. Juracy, A. de Morais Amory, and F. G. Moraes, "A fast, accurate, and comprehensive PPA estimation of convolutional hardware accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 12, pp. 5171–5184, Dec. 2022.

[8] N. Ali, J.-M. Philippe, B. Tain, and P. Coussy, "Exploration and generation of efficient FPGA-based deep neural network accelerators," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2021, pp. 123–128.

[9] M. Ferianc, H. Fan, D. Manocha, H. Zhou, S. Liu, X. Niu, and W. Luk, "Improving performance estimation for design space exploration for convolutional neural network accelerators," *Electronics*, vol. 10, no. 4, p. 520, Feb. 2021.

[10] A. Montgomerie-Corcoran, S. I. Venieris, and C.-S. Bouganis, "Power-aware FPGA mapping of convolutional neural networks," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2019, pp. 327–330.

[11] S. Goel, R. Kedia, R. Sen, and M. Balakrishnan, "EXPRESS: CNN EXecution time PREdiction for DPU DeSign space exploration," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2022, pp. 1–2.

[12] L. R. Juracy, M. T. Moreira, A. de Morais Amory, A. F. Hampel, and F. G. Moraes, "A high-level modeling framework for estimating hardware metrics of CNN accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 11, pp. 4783–4795, Nov. 2021.

[13] A. Mazouz and C. P. Bridges, "Automated offline design-space exploration and online design reconfiguration for CNNs," in *Proc. IEEE Conf. Evolving Adapt. Intell. Syst. (EAIS)*, May 2020, pp. 1–9.

[14] M. Hailesellasie, S. R. Hasan, and O. A. Mohamed, "MulMapper: Towards an automated FPGA-based CNN processor generator based on a dynamic design space exploration," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[15] C. Pham Quoc, X. Q. Nguyen, and T. N. Thinh, "Hardware/software co-design for convolutional neural networks acceleration: A survey and open issues," in *Proc. Int. Conf. Context-Aware Syst. Appl.* Cham, Switzerland: Springer, 2021, pp. 164–178.

[16] C. Pham-Quoc, X.-Q. Nguyen, and T. N. Thinh, "Towards an FPGA-targeted Hardware/Software co-design framework for CNN-based edge computing," *Mobile Netw. Appl.*, vol. 27, no. 5, pp. 2024–2035, Oct. 2022.

[17] The MathWorks, Inc. (2023). *Deep Learning HDL Toolbox*. Accessed: Sep. 26, 2023. [Online]. Available: https://www.mathworks.com/products/deep-learning-hdl.html

[18] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, M. Re, and S. Spanò, "FPGA-based road crack detection using deep learning," in *Proc. Int. Conf. Syst.-Integr. Intell.*, Genova, Italy. Cham, Switzerland: Springer, Sep. 2022, pp. 65–73.

[19] The MathWorks, Inc. (2023). *Deep Learning Processor IP Core Architecture*. Accessed: Sep. 26, 2023. [Online]. Available: https://www.mathworks.com/help/deep-learning-hdl/ug/deep-learning-processor-architecture.html

[20] The MathWorks, Inc. (2023). *Supported Networks, Layers, Boards, and Tools*. Accessed: Sep. 26, 2023. [Online]. Available: https://www.mathworks.com/help/deep-learning-hdl/ug/supported-networks-layers-boards-and-tools.html

[21] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

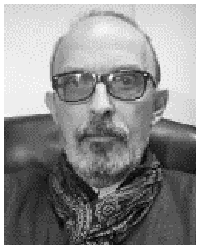
[22] S. Spanò, L. Canese, and G. C. Cardarilli, "Profiling of CNNs using the MATLAB FPGA-based deep learning processor," in *Proc. 17th Conf. Ph.D Res. Microelectron. Electron. (PRIME)*, Jun. 2022, pp. 121–124.

[23] The MathWorks, Inc. (2023). *AXI Manager*. Accessed: Nov. 8, 2023. [Online]. Available: https://www.mathworks.com/help/hdlverifier/axi-manager-xilinx.html

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

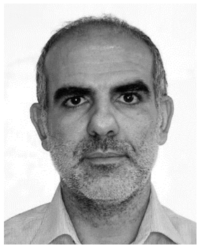
STEFANO BERTAZZONI received the Ph.D. degree in physics, specializing in microelectronics and telecommunications, from the Tor Vergata University of Rome. He is currently a Researcher with the Department of Electronic Engineering and a Professor of digital electronics with the Tor Vergata University of Rome. He is also the Founder of Xenta, a company specializing in the Yachting Industry, particularly in the development and integration of advanced marine control systems. His research interests include the development of neural networks with statistical learning functions and VLSI implementation, the investigation of electronic device performance in radiation environments, and the development of acquisition and data processing systems for optical RADARs.

LORENZO CANESE received the M.S. degree (summa cum laude) in electronic engineering from the Tor Vergata University of Rome, in 2020, where he is currently pursuing the Ph.D. degree in electronic engineering. His research interests include machine learning, swarm intelligence, ASIC/FPGA hardware design, and the design and digital implementation of multi-agent reinforcement learning algorithms.



GIAN CARLO CARDARILLI (Life Member, IEEE) was born in Rome, Italy. He received the Laurea degree (summa cum laude) from Sapienza Università di Roma, in 1981. Since 1984, he has been with the Tor Vergata University of Rome, where he is currently a Full Professor of digital electronics and electronics for communication systems. From 1992 to 1994, he was with the University of L'Aquila. From 1987 to 1988, he was with the Circuits and Systems Team, EPFL, Lausanne, Switzerland. He works in the field of computer arithmetic and its application to the design of fast signal digital processors. He has also regular cooperation with companies, such as Alcatel Alenia Space, Italy; STM, Agrate Brianza, Italy; Micron, Italy; and Selex S.I., Italy. His research interests include VLSI architectures for signal processing and IC design. In this field, he has published more than 160 papers in international journals and conferences. His scientific interest includes the design of special architectures for signal processing.

LUCA DI NUNZIO (Member, IEEE) received the master's degree (summa cum laude) in electronics engineering and the Ph.D. degree in systems and technologies for space from the Tor Vergata University of Rome, in 2006 and 2010, respectively. He is currently an Adjunct Professor with the Digital Electronics Laboratory, Tor Vergata University of Rome, and an Adjunct Professor of digital electronics with Guglielmo Marconi University. He has experience with several companies in the fields of electronics and communications. His research interests include reconfigurable computing, communication circuits, digital signal processing, and machine learning.



ROCCO FAZZOLARI received the master's degree in electronic engineering and the Ph.D. degree in space systems and technologies from the Tor Vergata University of Rome, Italy, in 2009 and 2013, respectively. He is currently a Postdoctoral Fellow and an Assistant Professor with the Department of Electronic Engineering, Tor Vergata University of Rome. He works on hardware implementation of high-speed systems for digital signals processing, machine learning, the array of wireless sensor networks, and systems for data analysis of acoustic emission (AE) sensors (based on ultrasonic waves).

MARCO RE (Member, IEEE) received the Ph.D. degree in microelectronics. He is currently an Associate Professor with the Tor Vergata University of Rome, where he teaches digital electronics and hardware architectures for DSP. He was awarded two NATO fellowships with Cadence Berkeley Laboratories, University of California at Berkeley, as a Visiting Scientist. He has been awarded the Otto Moensted Fellowship as a Visiting Professor with the Technical University of Denmark. He collaborates in many research projects with different companies in the field of DSP architectures and algorithms. He is the author of about 200 papers in international journals and international conferences. His main research interests include low-power DSP algorithms architectures, hardware-software codesign, fuzzy logic and neural hardware architectures, low-power digital implementations based on non-traditional number systems, computer arithmetic, and CAD tools for DSP. He is a member of the Audio Engineering Society (AES). He is also the Director of a Master in Audio Engineering with the Department of Electronic Engineering, Tor Vergata University of Rome.

SERGIO SPANÒ received the bachelor's, master's, and Ph.D. degrees (summa cum laude) in electronic engineering from the Tor Vergata University of Rome, in 2015 and 2018, respectively. Since 2022, he has been an Adjunct Professor with the Tor Vergata University of Rome, where he is currently a Postdoctoral Research Fellow. In addition, he is an Adjunct Professor with the "Guglielmo Marconi" University of Rome. He has several industrial work experiences in the fields of space and telecommunications. His research interests include digital signal processing, machine learning, the IoT, the development of telecommunication systems, and the implementation of machine learning accelerators for embedded and low-power systems.

Open Access funding provided by 'Università degli Studi di Roma ''Tor Vergata'' within the CRUI CARE Agreement