Received December 21, 2021, accepted January 2, 2022, date of publication January 12, 2022, date of current version January 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3142292

Approximate Down-Sampling Strategy for Power-Constrained Intelligent Systems

FANNY SPAGNOLO[1], (Member, IEEE), STEFANIA PERRI[2], (Senior Member, IEEE), AND PASQUALE CORSONELLO[1], (Member, IEEE)
[1]Department of Informatics, Modeling, Electronics and Systems Engineering, University of Calabria, 87036 Rende, Italy
[2]Department of Mechanical, Energy and Management Engineering, University of Calabria, 87036 Rende, Italy

Corresponding author: Pasquale Corsonello (p.corsonello@unical.it)

This work was supported in part by PON Ricerca & Innovazione - Ministero dell'Università e della Ricerca (Grant 1062_R24_INNOVAZIONE).

ABSTRACT In modern power constrained applications, as with most of those belonging to the Internet-of-Things world, custom hardware supports are ever more commonly adopted to deploy artificial intelligence algorithms. In these operating environments, limiting the power dissipation as much as possible is mandatory, even at the expense of reduced computational accuracy. In this paper we propose a novel prediction method to identify potential predominant features in convolutional layers followed by down-sampling layers, thus reducing the overall number of convolution calculations. This approximation down-sampling strategy has been exploited to design a custom hardware architecture for the inference of Convolutional Neural Network (CNN) models. The proposed approach has been applied to several benchmark CNN models and we achieved an overall energy saving of up to 70% with an accuracy loss lower than 3%, with respect to baseline designs. Performed experiments demonstrate that, when adopted to infer the Visual Geometry Group-16 (VGG16) network model, the proposed architecture implemented on a Xilinx Z-7045 chip and on the STM 28nm process technology dissipates only 680 and 21.9 mJ/frame, respectively. In both cases, the novel design overcomes several state-of-the-art competitors in terms of energy-accuracy drop product.

INDEX TERMS Approximate computing, convolutional neural networks, low-power hardware architectures, pooling layers.

I. INTRODUCTION

In the last few years, the inference of intelligent systems on low-end Internet-of-Things (IoT) mobile devices has attracted a lot of attention. The possibility of performing complex tasks "on the edge" offers significant advantages in terms of response latency, security and energy, since there is no need to transfer huge amounts of data to the backend through energy-hungry wireless transmissions. The deployment of such intelligent systems most often relies on deep learning and machine learning models, which have been proved effective in many fields of application, such as smart cities [1], Industry 4.0 [2] and cybersecurity [3].

Convolutional Neural Networks (CNNs) are a meaningful example of deep learning algorithms suitable to solve complex tasks, such as object detection and classification [4], speech recognition [5] and other human activities recognition [6]. State-of-the-art CNNs [7]–[9] exploit a high number of cascaded convolutional layers, interleaved by auxiliary layers that implement non-linear activations and down-sampling. Very deep models can require hundreds of millions of operations and intensive memory accesses that hinder their deployment on edge platforms. Therefore, inspired by the observation that small inaccuracies can be tolerated in the aforementioned applications, significant efforts have been focused in the recent past on conceiving methods able to reduce overall computational complexity and energy dissipation at the expense of achieved accuracy [10]–[24]. Table 1 provides a synthetic sketch of this scenario as given by a few recent papers. In several cases, quite significant gains are achieved with average accuracy penalties of about 6-7%.

Among the techniques summarized in Table 1, data quantization [10]–[13] and pruning [14] are certainly the most popular, since they allow approximating the input operands without modifying the basic operations (i.e. multiplications and accumulations) involved in a convolutional layer. As an example, the quantization strategy proposed in [13] reduces the input data to a much smaller set of values that

The associate editor coordinating the review of this manuscript and approving it for publication was Ilaria De Munari.

TABLE 1. Impact of approximation strategies on accuracy and computational complexity over floating-point baseline.

	Strategy	Model	Gain (%)	Top-1 Accuracy loss (%)
[11]	Quantization 8b weights	Alexnet	85.3†	3.22
[11]	Quantization 1b weights	Alexnet	94†	6.4
[12]	Cascaded Quantization	VGG16	NA	5
[13]	Input aware Quantization	CNN for digit classification	70†	2.5
[14]	Pruning	VGG16	71†	5
[15]	Weights reuse	LeNet-5	NA	7.76
[16]	Approx. Multipliers	Xception	83†	11.5
[17]	Approx. inference system (AxIS)	ResNet	70†	7.8
[18]	MAC skipping	LeNet-5	NA	8
[19]	MAC skipping by zero prediction	VGG16	75.5†	3.1
[21]	Pooling Aware Convolution	Alexnet	21.9‡	1.9
[21]	Pooling Aware Convolution	Alexnet	31.4‡	2.9

† Energy saving ‡ Reduction in the number of operations and/or memory accesses.

can be easily processed by replacing the energy-hungry arithmetic units with small and energy-efficient look-up-tables.

It is worth noting that, although quantizing 32-bit activations and filters to low precisions (e.g. 8-bit) is a common practice in designing CNN accelerators, the energy saving-accuracy loss ratio achievable in large CNNs, like VGG16, is typically lower than 30 [25]. Therefore, often, alternative approaches, including data reuse [15] and approximate computing [16]–[24], are adopted in conjunction with data quantization to further expand the design space exploration for the specific application.

Some of the referenced techniques [16]–[18] adopt operation-level approximations, while others [19]–[24] exploit data dependency across convolutional and auxiliary layers. In particular, the strategies presented in [19] and [20] are based on detection algorithms able to predict negative feature map values. Taking into account that negative values are zeroed when passing through a non-linear activation layer, such as the Rectified Linear Unit (ReLU), the multiply-and-accumulate (MAC) required to compute these values are skipped, thus reducing the overall computational load. A similar observation inspired the approaches proposed in [21]–[24], which save energy by reducing the computational complexity of convolutional layers followed by a pooling layer through partial computations of input operands.

This paper presents a new approximation method conceived to reduce the energy dissipation of CNNs inference on low-end IoT mobile devices. The proposed methodology is inspired by the observation that a certain number of values in the feature maps outputted by a convolutional layer will be discarded after down-sampling. The main contributions of this research are as follows:

- A novel prediction algorithm is introduced to predict potential predominant features, thus reducing the overall number of computations across the convolutional layers.
- A custom hardware architecture suitable to perform inference of CNNs with the proposed approximation method is presented.
- For purposes of comparison, a baseline hardware architecture implementing the inference of accurate and full-precision CNNs has been designed. Experimental results, obtained by Field-Programmable-Gate-Array (FPGA) and Application Specific Integrated Circuit (ASIC) implementations, highlight that the novel strategy achieves an overall energy saving of up to ~70%, at the expense of 3%, 0.4% and 2.2% accuracy loss for the LeNet-5, VGG16 and [26] CNN models, respectively.
- A comparison between the proposed VGG16 accelerator and state-of-the-art approximate designs is also presented. In this case, the new architecture experiences an energy-accuracy drop product up to 13.6 times lower than the competitors.

The remainder of this paper is structured as follows: Section II provides a brief background and motivations; Section III introduces the novel approximation method and a hardware architecture purpose-designed to operate as proposed here; results obtained from the comparison with prior works, in terms of classification accuracy, energy consumption, speed performances and area occupancy, are provided in Section IV; finally, conclusions are drawn in Section V.

II. BACKGROUND AND MOTIVATIONS

CNNs are typically composed of three different layer topologies. The convolutional layer is the computational centric element and massively performs multiply-and-accumulate (MAC) operations on 3D input data. More in detail, a convolutional layer receives a set of M input feature maps (*ifmaps*) with size $W_{in} \times H_{in}$ and applies on them $Mk \times k$ filters to extract features. The M results are summed-up by a pixel-wise addition, thus obtaining the so-called output feature map (*ofmap*). In order to extrapolate features at different levels, each convolutional layer usually produces multiple *ofmaps* that are transferred to the subsequent layer implementing a non-linear activation function, such as the Rectified Linear Unit (ReLU).

The depth of the model, the size of the input image to be classified, the size and the number of filters to be processed, can make the tasks of convolutional layers quite challenging. For this reason, almost all the state-of-the-art CNN algorithms use intermediate down-sampling layers that reduce the spatial dimension of the *ifmaps* as going deeper in the network. Among the several functions available for down-sampling, max-pooling [27] is certainly the most common. It is based on the criterion that only predominant features have to be propagated along the layers. A max-pooling layer uses a $k_P \times k_P$ sliding window that is moved across each feature map, in both horizontal and vertical dimensions, with a stride S_P. From each $k_P \times k_P$ patch of a feature map, the highest value is sent to the subsequent layer, whereas the remaining values are discarded.

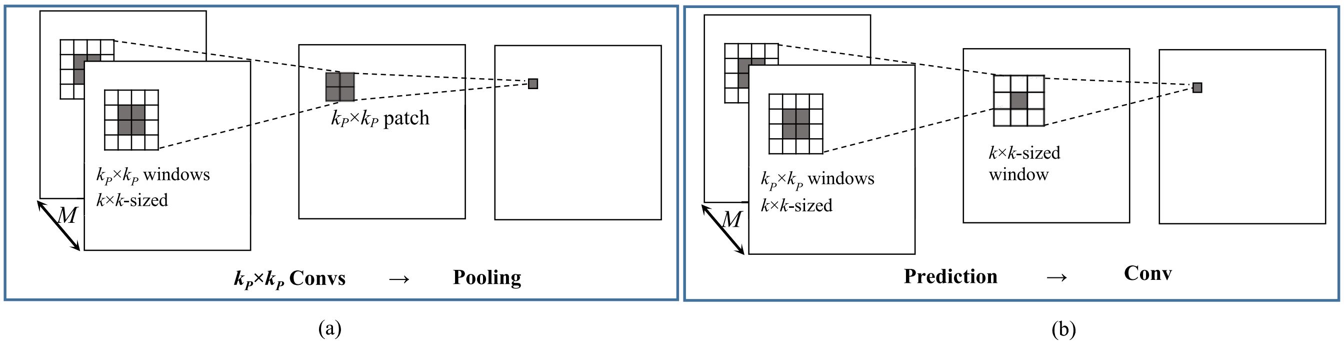

FIGURE 1. Description of (a) a conventional stack convolution-pooling; (b) the proposed computational method.

Figure 1(a) illustrates the usual operating scenario for a stack Convolution-Pooling, in which $Mk_P \times k_P$ convolutions are first computed on as many $k \times k$-sized windows and pixel-wise accumulated over the M input channels to output $k_P \times k_P$ values. The latter are then processed by the pooling stage to extract the final result. The above-described operations are defined in (1), where $s = (k\text{-}1)/2$ is the radius of the filter W.

$$ofmap\,(i, j) = \max_{\substack{r\in[0,kp-1]\\ c\in[0,kp-1]}} \left\{ \sum_{ch=0}^{M-1} \sum_{kx=-s}^{s} \sum_{ky=-s}^{s} [ifmap\ (i+r+kx, j +c + ky, ch) * W\,(kx + s, ky + s, ch)] \right\} \quad (1)$$

From Figure 1(a) it can be observed that the convolutional layer must also compute the values discarded by the pooling layer, thus wasting both time and energy. If predominant feature map values were predictable, many computations could be avoided and the computational cost of a given layer could be made $k_P \times k_P$ times lower. This strategy is particularly advantageous for all the CNNs that adopt Convolution-Pooling stacks. Apart from the well-known models LeNet-5, AlexNet, VGG16, and GoogleNet, also custom models oriented to specific applications, like smart healthcare, home robotics and traffic monitoring [28]–[30], can benefit from this improvement.

To predict the output of the max-pooling layer, previous works [21]–[24] preliminarily compute MAC operations on a sub-portion of the inputs bits; then, according to the adopted strategy, they are able either to estimate [22], [23] or to compute [21], [24] the exact result. In [22] and [23], the output of the preliminary computations is used to perform the exact convolution only on the useful *ifmap* values, which can lead, in the most favorable scenario, to a misclassification rate higher than 14% and an average power consumption just 14% lower than the conventional approach shown in Figure 1(a). Conversely, the techniques demonstrated in [21] and [24] skip the MAC operations only when the partial preliminary computations have exactly revealed the maximum. Owing to this, they reduce the number of operations less than [22], [23] and introduce a latency overhead, but they provide the exact output feature values.

In [31], Kim *et al.* demonstrated an efficient accelerator architecture for Binary CNNs (BCNNs) that reduces the cycle count by skipping some redundant operations, without penalizing the accuracy. This positive property is achieved by exploiting the particular characteristics of BCNNs, where *fmaps* and filters are binarized, and MAC operations can be replaced by XNOR-popcounts. The results obtained in this way are then compared with a threshold to identify redundant elements that can be safely skipped. However, also in this case, the process is not deterministic; therefore, the cycle count reduction is not uniform along the layers.

III. THE PROPOSED METHOD

The strategy presented here is synthetically illustrated in Figure 1(b): the $Mk_P \times k_P k \times k$-sized windows are processed by approximate operations to predict the pooling output in the $k_P \times k_P$ patch; this prediction is then exploited to perform just one accurate convolution. It is worth noting that, although in the following we will refer to the max-pooling function, the novel strategy can be applied to any kind of Convolution-Pooling stack. The performed operations are analytically described in (2), where (r_m, c_m) is the location of the maximum predicted element within the $k_P \times k_P$ patch, whereas *apConv* is the novel function, defined in (3), introduced to perform approximate convolutions. The latter process P_{fmaps} and P_{filter} obtained by encoding the *ifmaps* and the filters coefficients, respectively, through the novel encoding process summarized in Algorithm 1 and detailed in the following.

$$ofmap^{new}\,(i, j) = \sum_{ch=0}^{M-1} \sum_{kx=-s}^{s} \sum_{ky=-s}^{s} [if\,map\ (i + r_m + kx, j +c_m + ky, ch) * W\,(kx + s, ky + s, ch)] \quad (2a)$$

$$(r_m, c_m) = \operatorname*{argmax}_{\substack{r\in[0,kp-1]\\ c\in[0,kp-1]}} \{apConv\,(r_m, c_m)\} \quad (2b)$$

$$apConv\,(x, y) = \sum_{ch=0}^{M-1} \sum_{kx=-s}^{s} \sum_{ky=-s}^{s} \left[P_{fmaps}\,(i + kx, j + ky, ch) *P_{filter}\,(kx + s, ky + s, ch)\right] \quad (3)$$

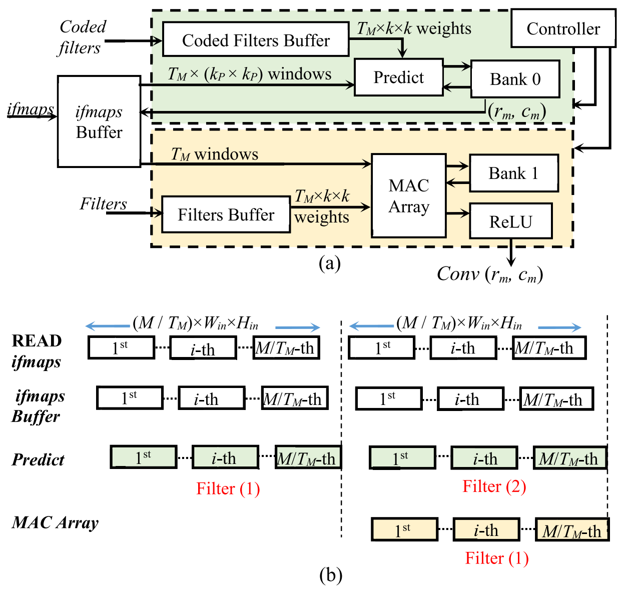

FIGURE 2. **The novel method: (a) hardware architecture; (b) timing diagram.**

It can be noted that, differently from the conventional computation described in (1), the new approach avoids redundant operations and reduces the number of precise convolutions to a quarter. Consequently, it can be expected that the energy dissipation due to the MAC operations will be reduced by at least ~75%. At a parity of parallelism level and adopted dataflow, this energy gain will be partially attenuated by the consumption caused by the prediction operations. Differently from [21]–[24], which exploit the straightforward truncation of input operands to perform preliminary approximate MAC operations, the novel prediction approach benefits from the *apConv* function defined in (3).

To introduce the hardware architecture based on the novel approach, let's consider a convolutional max-pooling stack, operating on $MW_{in} \times H_{in}$-sized *ifmaps*, and let's suppose that it can process T_M *ifmaps* channels in parallel. In this case, following a conventional computational scheme, to produce $NW_{out} \times H_{out}$-sized *ofmaps*, the M *ifmaps* have to be read N times to perform convolutions with as many $M \times k \times k$-sized filters.

The proposed method has been implemented by means of the purpose-designed folded architecture schematized in Figure 2(a). It is composed of two stages running concurrently, each one endowed with appropriate input and output buffers. The upper stage deals with the prediction step and selects the winner window in the $k_P \times k_P$ patch, whereas the lower stage uses the *MAC array* to compute the only convolution actually required on the selected window. It is worth noting that both the *Predict* module and the *MAC array* operate with the parallelism level T_M, therefore they require M/T_M iterations to process the whole *ifmaps* volume. The running of the proposed architecture is summarized in the timing diagram reported in Figure 2(b). It can be observed that the stages operate concurrently on different filters, thus,

Algorithm 1 The Proposed Encoding

```
1:  INPUT: 2D $IWin\ k \times k$, 2D $FWin\ k \times k$
2:  OUTPUT: $approxConv$
3:  $SizeSegFmap = R_{fmaps}/D_{fmaps}$; $SizeSegFilt = R_{filter}/D_{filter}$;
4:  $apConv = 0$;
5:  for $xwin = 0$ to $k$-$1$ step 1 do
6:      for $ywin = 0$ to $k$-$1$ step 1 do
7:          $w = FWin(xwin, ywin)$; $act = IWin(xwin, ywin)$;
8:          if($w$>0)do
9:              $sign = 2$;
10:         else
11:             $sign = 1$;
12:         for $vR_{fmaps} = 0$ to $D_{fmaps}$ do
13:             if($act = 0$)do
14:             $P_{fmaps} = 0$;
15:             break
16:             else if ($vR_{fmaps} \times SizeSegFmap \leq act < (vR_{fmaps}+1) \times SizeSegFmap$) do
17:                 $P_{fmaps} = vR_{fmaps}+1$;
18:                 break
19:
20:         for $vR_{filt} = 0$ to $\lceil D_{filter}/2 \rceil - 1$ do
21:             if($w = 0$)do
22:             $P_{filter} = 0$;
23:             break
24:             else if ($vR_{filt} \times SizeSegFilt \leq |w| < (vR_{filt}+1) \times SizeSegFilt$) do
25:                 $P_{filter} = (-1)^{sign} \times 2^{vRfilt}(-1)^{\mathrm{sign}} \times 2^{\mathrm{c}}$;
26:                 break
27:
28:         $apConv = apConv + (P_{fmaps} \times P_{filter})$;
```

in order to produce the N *ofmaps*, the *ifmaps* are read $N + 1$ times, which is just one time more than the conventional processing approach. Moreover, the proposed folded architecture requires both original and coded filters to be transferred in parallel from the external memory. The impact of such memory-related overhead is discussed later.

During the M/T_M iterations, the stages of the proposed architecture need to store provisional results in the memory banks *Bank 0* and *Bank 1*. The former stores $W_{in} \times H_{in}$ words with a reduced word-length, according to the implemented prediction logic, whereas the latter acts as an accumulation buffer, but, with only temporary results related to the winner window being stored, it is much smaller than a conventional buffer. When all the M *ifmaps* have been elaborated, the *Predict* module computes the maximum between the $k_P \times k_P$ values and it generates the indexes (r_m, c_m) that identify the window to be processed in the next stage.

The *Predict* module is structured as illustrated in Figure 3. It receives $T_M \times k_P \times k_P$ windows and $T_M k \times k$-sized filters.

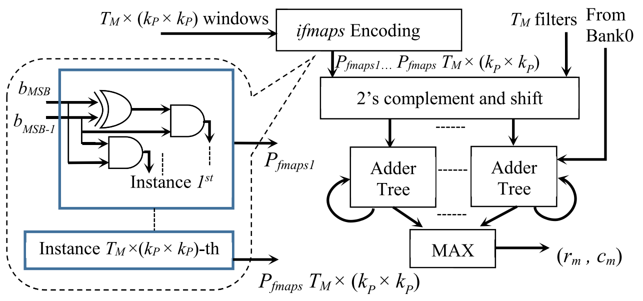
$T_M \times (k_P \times k_P)$ windows
ifmaps Encoding
T_M filters
From Bank0
b_{MSB}
b_{MSB-1}
Instance 1^{st}
P_{fmaps1}
P_{fmaps1}... P_{fmaps} $T_M \times (k_p \times k_p)$
2's complement and shift
Adder Tree
Adder Tree
MAX
(r_m, c_m)
Instance $T_M \times (k_P \times k_P)$-th
P_{fmaps} $T_M \times (k_p \times k_p)$

FIGURE 3. The *Predict* module. Input windows and coded filters have size $k \times k$.

It is worth noting that, since the filters are known a priori, their encoding process is performed offline. Conversely, the *ifmaps* are encoded through the simple combinatorial circuit illustrated in the inset of Figure 3. Such a circuit operates in parallel on the $T_M \times k_P \times k_P$ windows through as many identical instances.

The coded *ifmaps* are then left-shifted according to the coded filter coefficients and the obtained results are accumulated through the adder trees. The latter also receive provisional data produced in the previous step and resumed from the *Bank 0* memory. When all the *ifmaps* are processed, the indexes (r_m, c_m) are provided.

Algorithm 1 describes in detail the encoding process adopted for $k \times k$-sized *ifmaps* windows (*IWin*) and filters (*FWin*). To simplify the estimation process, the feature map values and the filter coefficients are properly coded by dividing their numeric ranges, namely R_{fmaps} and R_{filter}, into sub-ranges associated with progressive integer codes. The sub-ranges related to the unsigned *ifmaps* values are associated to the codes $P_{fmaps} = 0, 1, \ldots, D_{fmaps}$, with the code $P_{fmaps} = 0$ being reserved for zero value (lines 12-18). To this purpose, the combinatorial circuit, reported in the inset of Fig.3 for the case $D_{fmaps} = 4$, assigns the appropriate code to the *ifmap* value depending on its most significant bits. Conversely, the range R_{filter} of the signed filters coefficients is halved to spread the number of coded sub-ranges symmetrically between positive and negative values and to code the generic coefficient as the power-of-two integer number $P_{filter} = (-1)^{sign} \times 2^c$ (lines 20-27), with $sign = 2$ ($sign = 1$) for positive (negative) coefficients, and $c = 0, \ldots, \lceil D_{filter}/2 \rceil - 1$. Then, coded *ifmaps* and filters coefficients are multiplied by simple shift operations. These products are summed up to compute the approximate $k \times k$ convolution.

In order to better explain the proposed approach, Figure 4 provides an example of convolution between a 4×4 *ifmap* and the 3×3 filter *FWin*. As shown in Figure 4(a), when the conventional computation is performed, the four exact convolutions are computed on the 3×3 windows $IWin_0, \ldots, Iwin_3$ picked up from the *ifmap*. Supposing that the four exact convolution results are processed through a max-pooling stage with $k_P = 2$, the final output (i.e., 136.4) is given by $Iwin_1$. Figure 4(b) illustrates the result of the encoding process described in Algorithm 1 for *Fwin* and $Iwin_0, \ldots, Iwin_3$ when $R_{filters} = 2$, $D_{filters} = 8$ and $R_{fmaps} = 255$, $D_{fmaps} = 4$. In that case, being $SizeSegFilt = 0.25$ and $SizeSegFmap = 64$, the inputs are encoded as reported in P_{filter} and $P_{fmaps0}, \ldots, P_{fmaps3}$ respectively. Then, the generic approximate convolution $apConv_i$ is computed by accumulating the nine element-wise products between P_{filter} and P_{fmapsi}. The approximate results obtained in this way are compared to find the maximum value and to predict for which window the accurate convolution has to be performed. In the example of Figure 4, the proposed method correctly predicts the winner window (i.e. $Iwin_1$), thus allowing the number of accurate convolutions to be effectively reduced by 75%.

It is worth noting that the proposed architecture is able to perform also pure convolutional layers without the pooling. In this case, the prediction step is simply by-passed, while, as shown in Figure 2(a), the *ifmaps* Buffer directly feeds the *MAC array* with $T_M k \times k$ windows.

IV. EVALUATION AND DISCUSSION

To evaluate benefits and drawbacks of the proposed method, several experiments were performed on different CNNs. LeNet-5, VGG16 and the model in [26] were selected to process the benchmarks from the Modified National Institute of Standards and Technology (MNIST), Canadian Institute for Advanced Research (CIFAR10) and Street View House number (SVHN) [32] datasets. To perform a fair comparison, a set of baseline references were built up. The compared 32-bit architectures were implemented by using both the Xilinx XC7Z045 FPGA SoC and the STMicroelectronics 28nm Ultra-Thin Body and Buried oxide (UTBB) Fully-Depleted SOI (FDSOI) 1V process technology standard-cells library. Power analysis was performed considering both leakage and dynamic dissipation. For a realistic energy evaluation, which takes into account the actual switching activities, the complete system depicted in Figure 2(a) was fed with sample images from the benchmark datasets. The activity generated from post-implementation simulations was then used to output SAIF (Switching Activity Interchange Format) and VCD (Value Change Dump) data.

A. COMPARISON WITH THE BASELINE

The baseline designs accomplish the conventional stack depicted in Figure 1(a). For the purpose of a fair comparison in terms of throughput, their computational engines were made able to operate on $k_P \times k_P$ sub-windows in parallel with the circuit implementing the max-pooling function able to process one $k_P \times k_P$-sized patch at a time. To this purpose, all the baseline circuits characterized with $k_P = 2$ exploit four replicas of the same *MAC array* used in the new architecture. Tables 2 and 3 summarize the results obtained by comparing the proposed design, with $D_{filter} = 8$ and $D_{fmaps} = 4$, to the correspondent baseline system, at a parity of the parallelism level T_M and arithmetic precision (i.e., 32-bit fixed-point). It is worth noting that the architectures inferring the LeNet-5 and [26] models are tailored to support the same parallelism

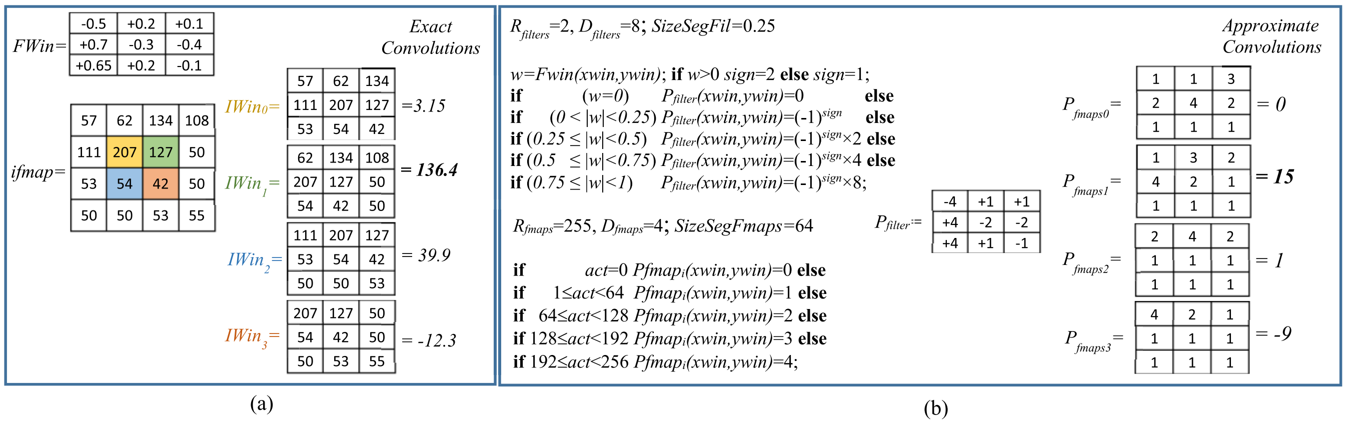
FWin=
-0.5 +0.2 +0.1
+0.7 -0.3 -0.4
+0.65 +0.2 -0.1
Exact Convolutions
ifmap=
57 62 134 108
111 207 127 50
53 54 42 50
50 50 53 55
IWin0=
57 62 134
111 207 127
53 54 42
=3.15
IWin1=
62 134 108
207 127 50
54 42 50
= 136.4
IWin2=
111 207 127
53 54 42
50 50 53
= 39.9
IWin3=
207 127 50
54 42 50
50 53 55
= -12.3
(a)
Rfilters=2, Dfilters=8; SizeSegFil=0.25
Approximate Convolutions
w=Fwin(xwin,ywin); if w>0 sign=2 else sign=1;
if (w=0) Pfilter(xwin,ywin)=0 else
if (0<|w|<0.25) Pfilter(xwin,ywin)=(-1)^sign else
if (0.25≤|w|<0.5) Pfilter(xwin,ywin)=(-1)^sign×2 else
if (0.5 ≤|w|<0.75) Pfilter(xwin,ywin)=(-1)^sign×4 else
if (0.75≤|w|<1) Pfilter(xwin,ywin)=(-1)^sign×8;
Pfilter=
-4 +1 +1
+4 -2 -2
+4 +1 -1
Rfmaps=255, Dfmaps=4; SizeSegFmaps=64
if act=0 Pfmapi(xwin,ywin)=0 else
if 1≤act<64 Pfmapi(xwin,ywin)=1 else
if 64≤act<128 Pfmapi(xwin,ywin)=2 else
if 128≤act<192 Pfmapi(xwin,ywin)=3 else
if 192≤act<256 Pfmapi(xwin,ywin)=4;
Pfmaps0=
1 1 3
2 4 2
1 1 1
= 0
Pfmaps1=
1 3 2
4 2 1
1 1 1
= 15
Pfmaps2=
2 4 2
1 1 1
1 1 1
= 1
Pfmaps3=
4 2 1
1 1 1
1 1 1
= -9
(b)

FIGURE 4. Example of 3 × 3 convolution between *FWin* and *ifmap*: (a) exact computations; (b) computations performed by Algorithm 1.

TABLE 2. Comparison with the baseline on FPGA.

	LeNet-5 (T_M=2)		[26] (T_M=2)		VGG16 (T_M=4)	
	Baseline	New	Baseline	New	Baseline	New
LUTs	147604	41932	147604	41932	159580	43148
FFs	101740	25798	101740	25798	10444	2860
BRAMs (18kbit)	8	6	10	8	10	8
F_{MAX} (MHz)	50	50	50	50	62	62
Inference Time (ms)	0.28	0.3	18.3	18.8	139.4	140.2
Energy (mJ/frame)	0.34	0.103	22.6	6.5	55.7	16.2

TABLE 3. Comparison with the baseline on ASIC (F_{MAX} = 500MHz).

	LeNet-5 (T_M=2)		[26] (T_M=2)		VGG16 (T_M=4)	
	Baseline	New	Baseline	New	Baseline	New
Area (mm^2)	0.64	0.2	0.64	0.2	0.47	0. 16
On-chip Memory (kB)	9.18	5.35	12	7	12	7
Inference Time (ms)	0.028	0.03	1.83	1.88	17.4	17.5
Energy (μJ/frame)	2.2	0.96	143.5	60.7	991.8	449.7

TABLE 4. Extra energy estimation.

	LeNet-5	[26]	VGG16
Energy overhead (μJ/frame)	0.0627	1.31	195

TABLE 5. Top-1and Top-5 accuracy (%).

	LeNet-5 (MNIST)		[26] (CIFAR10)		VGG16 (CIFAR10)		VGG16 (SVHN)	
	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1
FP	100	99.4	98.3	74.2	99	84.4	99.8	93.2
New4	99.8	96.2	98	70	99	79	n.a.	n.a.
New8	99.9	96.2	98	72	100	80.2	100	93
New16	99.9	96.2	97.8	71.3	100	81	100	93
New32	99.9	96.4	97.9	71.8	100	82	100	93
New64	99.9	96.3	97.8	72	100	84	99.6	92.8

n.a.: not applicable

level and kernel size. Therefore, they differ from each other just for the size of the memory banks and the *ifmaps* Buffer. At a glance, it is quite evident that the adopted approximate strategy leads to significant improvements in terms of power consumptions and area occupancy for both FPGA- and ASIC-based implementations. As an example, in the former case, the new circuit saves more than 71% of LUTs and FFs and reduces the energy consumption by at least 70%. These results are intrinsically due to the novel computational paradigm, which allows reducing the number of exact MAC operations by the $k_P \times k_P$ factor. The slightly more complex *ifmaps* buffer structure and the additional memory banks, above discussed, account for only 5% of the global energy consumed by the proposed system. It is worth noting that, in these first experiments, to make the results more general, DSP slices were not used in the FPGA synthesis.

The inference times reported in Tables 2 and 3 take into account the additional *ifmaps* read round, consisting of $(M/T_M) \times W_{in} \times H_{in}$ clock cycles. Estimating the energy dissipation overhead caused by these extra external memory accesses is not an easy task. Indeed, the DRAM energy consumption depends on the memory micro-architecture, its physical floorplan, the process characteristics and the toggle data rate [33]. Nevertheless, considering the behavior of the latest low-power *High Bandwidth Memory* (HBM2), we estimated the overheads reported in Table 4. The latter clearly shows that the extra energy due to the additional memory accesses is negligible for all the FPGA-based and most of the ASIC-based implementations. It is worth noting that, due to the larger *ifmap* volume involved in each layer, the energy consumed by the VGG16 network for memory data transfers becomes much more significant. However, also in this case, the energy saving achieved by the proposed design is higher than the overhead caused by the extra memory activities.

Moreover, it is worth noting that the need to upload also the encoded filters from the external memory increases the memory bandwidth by just 6%, thus keeping the overall requirement well below the effective performances of commercial memories.

Classification accuracies and energy requirements can be traded-off exploiting different coding ranges for filters and *ifmaps*. Since the former are encoded offline, D_{filter} can be chosen efficiently examining the statistic distribution of the filter coefficients. Conversely, D_{fmaps} could be runtime configured moving deep into the CNN model. However, in the following, the proposed architecture has been characterized

TABLE 6. Comparison with prior works on VGG16 (FPGA).

	Strategy	Device	Precision	F_{MAX} (MHz)	LUTs	FFs	DSPs	Memory (Mbit)	Energy (mJ/frame)	Energy Efficiency (GOPS/W)	Energy-Accuracy drop Product
[12]	Cascaded quantization	Z-7045	[4b,7b]* fixed-point	150	~177k	NA	900	~1.1	NA	NA	NA
[34]†	Pruning	Virtex-7	16b fixed-point	100	103k	107.6k	471	7.85	NA	23.2	NA
[35]	Fixed quantization	Z-7100	16b fixed-point	60	229k	107k	128	13.5	1170	27.4	936
[36]	Fixed quantization	Z-7020	16b fixed-point	50 (conv)	35k	43k	68	1.5	903.6	19.1⁺	3704.7
New64	Coding inputs for preliminary estimation of pooling result	Z-7045	32b fixed-point	62	17.7k	11.8k	288	6.06	680	45.8	272

†Targeting 32×32 input images.*Refers to the [LPU, HPU] modules in [9]. ⁺Obtained for GOPS=2×DSPs× F_{MAX}

TABLE 7. Comparison with prior works on VGG16 (ASIC- in brackets results scaled to 28 *nm*).

	Strategy	Technology, Supply voltage	Precision	F_{MAX} (MHz)	Area (mm^2)	On-chip SRAM (kB)	Energy (mJ/frame)	Energy-Accuracy drop Product
[19]	MAC skipping by zero prediction	65nm, 1V	16b fixed-point	200 (377)	14.01 (3.5)	NA	NA	NA
[24]	MAC truncation for preliminary identification of pooling result	40nm, 0.9V	12b fixed-point	400 (439)	9 (3)	339.5	25 (20.5)	16.4
[38]	Fixed quantization	28nm, 1V	16b fixed-point	200	1.87	144	37.1	NA
New64	Coding inputs for preliminary estimation of pooling result	28nm, 1V	32b fixed-point	500	1.07	343	21.9	8.76

considering $D_{filter} = 8$ and D_{fmaps} ranging between 4 and 64. Table 5 reports the accuracy results, in term of Top-1 and Top-5 percentages, for the Full-precision (i.e., 32-bit floating-point) and the New x (with $D_{fmaps} = x$) implementations of the benchmark networks. It must be noted that, due to the limited dynamic range on RGB components observable in the SVHN benchmark images, the $D_{fmaps} = 4$ configuration is not applicable. The obtained accuracy penalties, which are coherent with data reported in Table 1 for alternative approximation approaches, are the more than reasonable price to pay for reducing the energy requirements by up to 71% with respect to the baseline counterparts. It is worth noting that when applied to the VGG16, the energy saving-accuracy loss ratio achieved by the proposed approximate methodology is up to 4 times higher also than a conventional architecture using a more aggressive 8-bit quantization on both weights and activations [25].

Finally, Figure 5 illustrates the percentage improvements achieved for the ASIC implementations in terms of area and power saving versus D_{fmaps}. While D_{fmaps} does not significantly affects the silicon area, the smaller D_{fmaps} the more power dissipation benefits. Table 5 and Figure 5 show that the best trade-off between the accuracy and the energy saving is achieved with $D_{fmaps} = 32, 8, 64$ for LeNet-5, [26] and VGG16, respectively.

B. COMPARISON WITH PRIOR WORKS ON VGG16

To further validate the proposed approach, the architecture for accelerating the VGG16 model has been compared with several state-of-the-art accelerators adopting some of the approximate strategies listed in Table 1. Tables 6 and 7 collect the results related to FPGA and ASIC implementations,

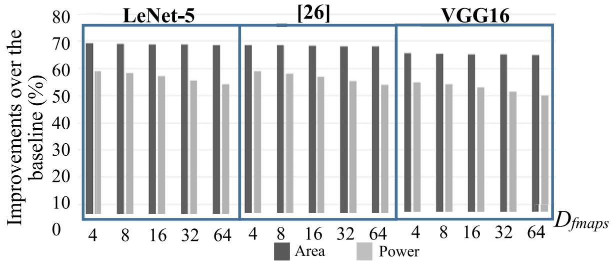

FIGURE 5. Power and area saving for the proposed ASIC-based architectures at different D_{fmaps} configurations.

respectively. Data reported in the tables are extracted from original papers. For the sake of comparison, in this case, the novel architecture has been synthesized to process 224 × 224 input images. Moreover, to perform the arithmetic computations efficiently, the FPGA design has been made able to exploit the DSP units available within the device. From Table 6, it can be seen that, despite the higher precision, the proposed technique leads to the lowest energy dissipation per frame and the best energy efficiency expressed in terms of Giga operations per second per watt (GOPS/W). Indeed, the proposed architecture consumes up to ∼42% and ∼25% less energy than the 16b fixed quantization approaches demonstrated in [35] and [36]. Moreover, the strategy proposed here achieves an energy efficiency ∼49% higher than that reached by the architecture in [34]. This energy saving is obtained without penalizing the achieved accuracy: the improvement reached by the New64 implementation over its corresponding baselines in terms of the product between the consumed energy and the accuracy drop is up to 13.6 times higher than state-of-the-art competitors.

Table 7 shows results for ASIC implementations. Performance parameters scaled using the method presented in [37] are also reported in brackets. At a parity of the process technology, the technique proposed here achieves an energy per frame ∼41% lower than [38]. From scaled results, it is clear that both [19] and [24] span over a silicon area ∼3× larger than that required by the proposed architecture. A more interesting consideration arises from the comparison with the approach proposed in [24]. The latter skips redundant MAC operations by recursively applying approximate computing, until the output of the max pooling layer is identified. With the number of skipped MAC operations being not deterministic (indeed it is pattern dependent), such a technique can lead to latency overheads, which obviously affect the speed performances. As reported in Table 7, the energy consumed by [24] is only ∼6% lower than the proposed design at 32b fixed-point, but it achieves a maximum clock frequency ∼12% lower. Moreover, since [24] experiences a Top-1 accuracy drop doubled with respect to the proposed New64 implementation, its Energy-Accuracy drop product is ∼47% higher.

V. CONCLUSION

In this paper we demonstrated a novel approximate down-sampling method for the efficient design of CNN accelerators in energy-constrained systems. It adopts a quite simple yet effective encoding process on the *ifmaps* and the filters coefficients to reduce the number of computations wherever a convolutional layer is followed by a down-sampling layer. The proposed approach has been characterized by using both FPGA and ASIC technologies. In the former case, it has been proved that this strategy allows the energy-per-frame to be reduced up to ∼71%, with a Top-1 accuracy penalty of only 0.4%. ASIC prototypes achieved an energy-per-frame reduction up to ∼58%, maintaining the original inference time. Specific architectures based on the proposed approach have been implemented for comparison purpose with several state-of-the-art competitors. They infer the VGG16 CNN with 224 × 224 input image size. The FPGA-based prototype running at 62 MHz clock frequency dissipates only 680 mJ/frame, reaches 45.8 GOPS/W and shows the lowest Energy-accuracy drop product. Among compared ASIC accelerators, the proposed structure spans over 1.07 mm^2 of silicon area and consumes 21.9 mJ/frame, which is only ∼7% more than [24], but with an energy accuracy drop product ∼47% lower. Overall, obtained results demonstrated that the proposed strategy achieves an energy/accuracy trade-off more favorable than most of the state-of-the-art approaches referenced in Table 1. A framework for automated run-time re-configuration of D_{filter} and D_{fmaps} could be an interesting future extension of this research work.

REFERENCES

[1] T. M. Ghazal, M. K. Hasan, M. T. Alshurideh, H. M. Alzoubi, M. Ahmad, S. S. Akbar, B. Al Kurdi, and I. A. Akour, "IoT for smart cities: Machine learning approaches in smart healthcare—A review," *Future Internet*, vol. 13, no. 8, p. 218, Aug. 2021.

[2] M.-Q. Tran, M. Elsisi, K. Mahmoud, M.-K. Liu, M. Lehtonen, and M. M. F. Darwish, "Experimental setup for online fault diagnosis of induction machines via promising IoT and machine learning: Towards industry 4.0 empowerment," *IEEE Access*, vol. 9, pp. 115429–115441, 2021.

[3] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[5] D. Palaz, M. M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 4295–4299.

[6] K. Xia, J. Huang, and H. Wang, "LSTM-CNN architecture for human activity recognition," *IEEE Access*, vol. 8, pp. 56855–56866, 2020.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015, pp. 398–406.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[10] C. Y. Lo and C.-W. Sham, "Energy efficient fixed-point inference system of convolutional neural network," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 403–406.

[11] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar, and S. Reda, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1478–1483.

[12] A. Kouris, S. I. Venieris, and C.-S. Bouganis, "Cascade^{CNN}: Pushing the performance limits of quantisation in convolutional neural networks," in *Proc. 28th Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2018, pp. 155–162.

[13] A. Raha and V. Raghunathan, "qLUT: Input-aware quantized table lookup for energy-efficient approximate accelerators," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–23, Oct. 2017.

[14] M.-A. Maleki, A. Nabipour-Meybodi, M. Kamal, A. Afzali-Kusha, and M. Pedram, "An energy-efficient inference method in convolutional neural networks based on dynamic adjustment of the pruning level," *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 6, pp. 1–20, Aug. 2021.

[15] M. F. Tolba, H. T. Tesfai, H. Saleh, B. Mohammad, and M. Al-Qutayri, "Deep neural networks based weight approximation and computation reuse for 2-D image classification," 2021, *arXiv:2105.02954*.

[16] M. S. Kim, A. A. D. B. Garcia, H. Kim, and N. Bagherzadeh, "The effects of approximate multiplication on convolutional neural networks," *IEEE Trans. Emerg. Topics Comput.*, early access, Jan. 12, 2021, doi: 10.1109/TETC.2021.3050989.

[17] S. K. Ghosh, A. Raha, and V. Raghunathan, "Approximate inference systems (AxIS): End-to-end approximations for energy-efficient inference at the edge," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2020, pp. 7–12.

[18] F. Alantali, Y. Halawani, B. Mohammed, and M. Al-Qutayri, "SLID: Exploiting spatial locality in input data as a computational reuse method for efficient CNN," *IEEE Access*, vol. 9, pp. 57179–57187, 2021.

[19] C. Kim, D. Shin, B. Kim, and J. Park, "Mosaic-CNN: A combined two-step zero prediction approach to trade off accuracy and computation energy in convolutional neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 4, pp. 770–781, Dec. 2018.

[20] V. Akhlaghi, A. Yazdanbakhsh, K. Samadi, R. K. Gupta, and H. Esmaeilzadeh, "SnaPEA: Predictive early activation for reducing computation in deep convolutional neural networks," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 662–673.

[21] A. Sayal, S. Fathima, S. T. Nibhanupudi, and J. P. Kulkarni, "COMPAC: Compressed time-domain, pooling-aware convolution CNN engine with reduced data movement for energy-efficient AI computing," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2205–2220, Jul. 2021.

[22] T. Ujiie, M. Hiromoto, and T. Sato, "Approximated prediction strategy for reducing power consumption of convolutional neural network processor," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 870–876.

[23] M. Ahmadi, S. Vakili, J. M. P. Langlois, and W. Gross, "Power reduction in CNN pooling layers with a preliminary partial computation strategy," in *Proc. 16th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2018, pp. 125–129.

[24] M. Kim and J.-S. Seo, "Deep convolutional neural network accelerator featuring conditional computing and low external memory access," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Mar. 2020, pp. 1–4.

[25] G. Yuan, P. Dong, M. Sun, W. Niu, Z. Li, Y. Cai, J. Liu, W. Jiang, X. Lin, B. Ren, X. Tang, and Y. Wang, "Work in progress: Mobile or FPGA? A comprehensive evaluation on energy efficiency and a unified optimization framework," in *Proc. IEEE 27th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2021, pp. 493–496.

[26] A. Krizhevsky and G. Hinton. *Learning Multiple Layers of Features From Tiny Images*. Accessed: Jan. 13, 2022. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[27] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*, 2010, pp. 92–101.

[28] A. R. Lopez, X. Giro-I-Nieto, J. Burdick, and O. Marques, "Skin lesion classification from dermoscopic images using deep learning techniques," in *Proc. 13th IASTED Int. Conf. Biomed. Eng. (BioMed)*, Feb. 2017, pp. 49–54.

[29] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 421–436, Apr. 2018.

[30] H. Li, Y. Li, and F. Porikli, "DeepTrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1834–1848, Apr. 2016.

[31] T.-H. Kim and J. Shin, "A resource-efficient inference accelerator for binary convolutional neural networks," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 1, pp. 451–455, Jan. 2021.

[32] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, Dec. 2011, pp. 1–9.

[33] M. O'Connor, N. Chatterjee, D. Lee, J. Wilson, A. Agrawal, S. W. Keckler, and W. J. Dally, "Fine-grained DRAM: Energy-efficient DRAM for extreme bandwidth systems," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchit.*, Oct. 2017, pp. 41–54.

[34] W. Pang, C. Wu, and S. Lu, "An energy-efficient implementation of group pruned CNNs on FPGA," *IEEE Access*, vol. 8, pp. 217033–217044, 2020.

[35] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu, and T. Delbruck, "NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 644–656, Mar. 2019.

[36] T. Sledevič and A. Serackis, "mNet2FPGA: A design flow for mapping a fixed-point CNN to Zynq SoC FPGA," *Electronics*, vol. 9, no. 11, p. 1823, Nov. 2020.

[37] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, Jun. 2017.

[38] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2017, pp. 246–247.

FANNY SPAGNOLO (Member, IEEE) was born in Belvedere Marittimo, Cosenza, Italy, in April 1991. She received the master's degree in electronics engineering and the Ph.D. degree in information and communication technologies from the University of Calabria, Italy, in 2016 and 2019, respectively. In June 2016, she won a Research Grant funded by the Department of Informatics, Modeling, Electronics and System Engineering, University of Calabria. She is currently a Research Fellow at the University of Calabria. Her research interests include embedded systems design for real time video processing, high-performance reconfigurable circuits design, and energy efficient hardware implementation of deep neural networks.

STEFANIA PERRI (Senior Member, IEEE) was born in Cosenza, Italy, in April 1971. She received the master's degree in computer science engineering from the University of Calabria, Italy, in 1996, and the Ph.D. degree in electronics engineering from the University Mediterranea of Reggio Calabria, Italy, in 2000. In 1996, she joined the Department of Electronics, Computer Sciences and Systems, University of Calabria, as a Researcher Associate. In 2002, she was appointed as an Assistant Professor of electronics with the Department of Electronics, Computer Science and Systems, University of Calabria. In the summer 2004, she was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA, where she was appointed as an Adjunct Assistant Professor, for a period of four years, in 2005. In 2010, she was appointed as an Associate Professor of electronics with the Department of Electronics, Computer Sciences and Systems, University of Calabria. In 2017, she joined the Department of Mechanical, Energy and Management Engineering, University of Calabria. She is the coauthor of more than 140 technical articles and holds two patents in these fields. Her current research interests include QCA-based circuits, high-performance embedded systems, low-power design, VLSI circuits for image processing and multimedia, reconfigurable computing, and VLSI design. She serves on technical committees of several VLSI conferences and as a peer reviewer for several VLSI journals. She is an Associate Editor of the *Journal of Low Power Electronics and Applications* and *Sensors*.

PASQUALE CORSONELLO (Member, IEEE) was born in Cosenza, Italy, in May 1964. He received the master's degree in electronics engineering from the University of Naples Federico II, Naples, Italy, in 1988. He joined the Institute of Research on Parallel Computers, National Council of Research of Italy, Naples, where he was working on the design and modeling of electronic transducers for high precision measurement, receiving a post-graduate two-years grant. In 1992, he joined the Department of Electronics, Computer Science and Systems, University of Calabria, Rende, Italy, as a Research Associate. In 1997, he was appointed as an Assistant Professor of electronics with the Department of Electronics Engineering and Applied Mathematics, University of Reggio Calabria, Reggio Calabria, Italy, where he also worked as the Director of the Microelectronics Laboratory. In 2001, he was appointed as an Associate Professor of electronics and as the Chair of the Ph.D. Program in Electronics Engineering at the University of Reggio Calabria. In the summer 2004, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA, where he was appointed as an Adjunct Associate Professor, in 2005. He is currently a Full Professor of electronics at the Department of Informatics, Modeling, Electronics and System Engineering, University of Calabria. He has coauthored over 180 technical articles and holds two patents in these fields. His main research interests include embedded systems design, low-power design, VLSI architecture for image processing, and QCA-based circuits. He serves on technical committees of several VLSI conferences and as a peer reviewer for several VLSI journals. He served as the Editor-in-Chief for the *Journal of Low Power Electronics and Applications*. He is an Associate Editor-in-Chief of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS.