Always-ON Visual node with a Hardware-Software Event-Based Binarized Neural Network Inference Engine

Manuele Rusci
DEI, University of Bologna, IT
manuele.rusci@unibo.it

Davide Rossi
DEI, University of Bologna, IT
davide.rossi@unibo.it

Eric Flamand
GreenWaves Technologies, FR
eric.flamand@greenwaves-technologies.com

Massimo Gottardi
Fondazione Bruno Kessler, IT
gottardi@fbk.eu

Elisabetta Farella
Fondazione Bruno Kessler, IT
efarella@fbk.eu

Luca Benini
DEI, University of Bologna, IT
D-ITET, ETH Zurich, CH
luca.benini@unibo.it

ABSTRACT

This work introduces an ultra-low-power visual sensor node coupling event-based binary acquisition with Binarized Neural Networks (BNNs) to deal with the stringent power requirements of always-on vision systems for IoT applications. By exploiting in-sensor mixed-signal processing, an ultra-low-power imager generates a sparse visual signal of binary spatial-gradient features. The sensor output, packed as a stream of events corresponding to the asserted gradient binary values, is transferred to a 4-core processor when the amount of data detected after frame difference surpasses a given threshold. Then, a BNN trained with binary gradients as input runs on the parallel processor if a meaningful activity is detected in a pre-processing stage. During the BNN computation, the proposed Event-based Binarized Neural Network model achieves a system energy saving of 17.8% with respect to a baseline system including a low-power RGB imager and a Binarized Neural Network, while paying a classification performance drop of only 3% for a real-life 3-classes classification scenario. The energy reduction increases up to 8x when considering a long-term always-on monitoring scenario, thanks to the event-driven behavior of the processing sub-system.

KEYWORDS

always-on vision system, event-based sensing, binarized neural network, ultra-low power visual sensing

ACM Reference format:
Manuele Rusci, Davide Rossi, Eric Flamand, Massimo Gottardi, Elisabetta Farella, and Luca Benini. 2016. Always-ON Visual node with a Hardware-Software Event-Based Binarized Neural Network Inference Engine. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Convolutional neural networks achieve state-of-the-art performance on several recognition tasks, such as image classification, object detection and voice recognition. Despite the promising capabilities, the high computational cost is preventing their deployment on resource-constrained and ultra-low power devices. Recently, Binarized Neural Networks (BNNs) have emerged as a potential solution for pushing deep networks into battery-powered systems [5, 17]. The extreme 1-bit quantization of both the weights and activation layers reduces the memory requirement by 32x while still leading to a nearly state-of-the-art accuracy on several public dataset [5]. By reducing the internal precision to a single bit, the convolution operation, which is the most demanding kernel of a CNN inference algorithm, transforms into a bit-wise logic XNOR followed by a bit counting operation. This has enabled the deployment of deep networks into low-power programmable processors featuring 32bit bit-wise operations, also thanks to their flexibility and easy programming legacy [15]. However, due to the stringent requirement of always-ON sensors, a BNN inference engine needs to be coupled with power-optimized smart sensing devices to further reduce the system power consumption.

In the context of camera-based system design, the event-based sensing paradigm has shown significant benefits from an energy viewpoint by keeping the system in a low-power mode unless a notable amount of information, in the form of events, is detected on the sensor die [21]. In this paper, we present an Event-Based Binarized Neural Network that combines a BNN implementation on a software-programmable 4-core processor with an event-based binary acquisition scheme, aiming at enhancing the detection capabilities and, at the same time, matching the energy requirement of a low power visual sensor front-end. As opposed to previously presented BNN approaches, where RGB pixels are produced by a traditional camera and then binarized on the processing platform, we leverage in-pixel hardware processing circuits to produce the binary data to feed into the neural network. Every pixel is generated by thresholding the gradient computed over a three-pixel spatial mask [8]. This process naturally reflects the operation of a binarized pixel-wise convolution and can be seen as embedding the first convolutional layer within the image sensor die. Moreover, mixed-signal in-sensor processing allows to lower the power consumption of the imagers by more than 10× with respect to power-optimized off-the-shelf components [2] thanks to the reduced amount of data crossing the costly analog-to-digital border [13].

The main contributions of the paper are:

(1) The specification of an Event-Based Binarized Neural Network model, which fits the energy requirements and resource constraints of a deeply-embedded always-on visual sensing front-end.
(2) An optimized implementation of a BNN on a 4-core embedded processor.
(3) The energy evaluation of the proposed solution and the comparison with a baseline BNN model with RGB data input.

Conference'17, Washington, DC, USA
2016. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

To assess the performance of our Event-Based Binarized Neural Network solution, we collect real-world images belonging to three different classes and train a binarized neural network. The BNN inference model is implemented on a 4-core RISC processor optimized for energy-efficiency [19]. As an outcome, we demonstrate that during the classification task, the proposed solution reduces the system energy by 17.8% in reference to a frame-based baseline system that includes a low-power RGB imager and a traditional BNN approach, while paying only a 3% reduction of classification performance on a 3-classes scenario. Moreover, when considering a long-term monitoring application, the system leverages the event-based sensing scheme to reduce the start-up activity of the processor and to trigger a classification run upon the detection of a relevant event. This leads to an energy reduction of up to 8x with respect to the frame-based system featuring an RGB camera.

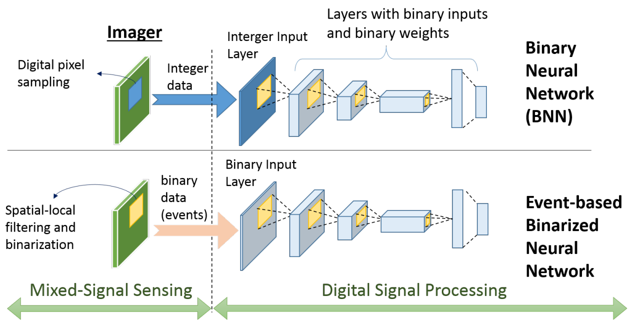
Imager
Digital pixel sampling
Integer data
Interger Input Layer
Layers with binary inputs and binary weights
Binary Neural Network (BNN)
Spatial-local filtering and binarization
binary data (events)
Binary Input Layer
Event-based Binarized Neural Network
Mixed-Signal Sensing
Digital Signal Processing

Figure 1: Comparison between a traditional BNN flow and the proposed Event-based BNN scheme, which exploits focal-plane processing for in-sensor binarization

2 RELATED WORK

Recent work on Convolutional Neural Networks (CNNs) has investigated the bit quantization to reduce the storage and computational costs of the inference task. As an extreme case, Binarized Neural Networks (BNNs) reduce the precision of both weights and activation neurons to a single-bit [5, 17]. As a distinctive feature, the binary quantization is not only applied during the forward pass, but also during the backward pass of the gradient descent algorithm, and acts as a sort of regularizer [16]. Hardware accelerators for highly-quantized NNs have been presented on FPGA [23], ASIC [3, 10] and neuromorphic brain-inspired chips such as Truenorth [6], trading the flexibility of general-purpose processors with highest performance and energy efficiency of specialized hardware. To lower the computational complexity of BNNs, a hardware-oriented kernel decomposition strategy is presented in [10], using clock-gating to reduce the energy cost of redundant convolutions. This is clearly effective but less prone to be implemented in software because it is weight-dependent and does not benefit from data spatial contiguity, which is exploited in this work to reduce the computation latency. Indeed, we target a low-power software-programmable platform as a digital processing unit for running the BNN classifier, also due to the lower development time with respect to application specific circuits and reconfigurable logic. A first software-based implementation of a binary network on resource-constrained devices is presented in [15] and it is based on an Arduino platform. Starting from this work, we optimize the implementation and extend it to a parallel ultra-low-power processor.

To address energy issues in the context of smart visual sensors, moving part of the computation to the sensor die results effective since it allows to lower the sensor-to-processor data flow and the workload of the digital signal processing [18]. The sensor front-end described in [12] leads to a maximum 64x reduction of ADC conversions by integrating a matrix-multiply-and-accumulate analog circuit before digitization. Pushing further the early-processing approach, the design of an analog convolutional image sensor is presented in [13]. Simulation results show a reduction of energy up to 84% when the first layers of the CNN are computed in the analog domain, together with the energy reduction on the digital processor. With respect to these solutions, the imager used in this work [8] still features in-sensor processing capabilities. Furthermore it provides a binarized output, which naturally reflects the behavior of a binarized convolutional layer.

A first study of applying deep learning approaches on the gradient-based data generated by the imager is presented in [9]. Authors focused on greyscale image reconstruction from gradient-based data while not targeting implementation on resource-constrained systems. More in details, we evaluate the combination of a BNN for classification purpose with an Event-Based sensing scheme, which has been demonstrated to be extremely efficient for continuous sensing thanks to the event-driven activation of the digital processor depending on the external context activity [21].

3 EVENT-BASED BNN

The Event-based Binarized Neural Network scheme is depicted in Fig. 1. The presented approach combines the paradigm of event-based sensing with the concept of Binarized Neural Networks. If compared with traditional BNN architectures that operates on 3-channels RGB images, the Event-Based BNN is fed by sensor data that has been pre-processed and binarized on the image sensor die. This latter exploits a pixel-wise hardwired spatial filtering operation: a per-pixel mixed-signal circuit computes the weighted gradient across a neighboring pixel mask. Following this, gradient values are binarized by thresholding. We refer to any of the asserted pixel, i.e. the ones with a gradient value greater than the threshold, as an event. The event-based paradigm allows to drastically reduce the sensor-to-processor bandwidth and memory footprint on the processor-side if compared to the baseline that transfers and processes RGB sensor data. The BNN is implemented on the digital signal processor and refers to the model presented by Courbariaux et al. [5]. Thanks to the input and weight binarization, any convolution reduces to a logic XNOR and a bit counting operation, resulting suitable to be implemented on ultra-low-power processors based on RISC architectures.

In this work, we refer to the imager prototype presented in [8] to investigate the Event-Based Binarized Neural Networks. The hardware in-sensor processing stands as the first stage of the inference pipeline. For any pixel location *PO*, the local gradient is computed with respect to the neighboring pixels, *PN* and *PE*, as illustrated in figure 2a. Assuming *PO* and *PN* to be respectively the more and the less exposed pixels to light, the gradient will be proportional to the voltage difference between *PO* and *PN* (fig. 2b). Figure 2c illustrates the circuit that implements this pixel-wise functionality. The output of the contrast block is binarized by means of the comparator *comp2*, considering a tunable level V_{th}. In the adopted prototype, this threshold is generally set close to 0V, to gain a higher sensitivity.

The sensor dispatches out only the asserted pixels, i.e. the ones with a positive value after thresholding. The data output is formatted as a stream of (x,y) coordinates, corresponding to the generated events. According to the event-based paradigm, the amount of transferred data is varying depending on the context activity.

A preliminary experiment to assess the capability of the approach is conducted by benchmarking against the CIFAR-10 dataset [11]. To simulate the in-sensor processing of our imager, we use a basic sensor model to convert RGB images of the dataset into the binary representation space. As a first approximation, the gradient contrast V_{EDGE} is computed as:

$$V_{EDGE} = \frac{\max(|p_E - p_O|, |p_N - p_O|)}{\max(p_E, p_O, p_N)} \tag{1}$$

where p_X is the greyscale value of the correspondent RGB pixel value. The binarization is formulated as:

$$V_O = sign(V_{EDGE} - V_{th}) \tag{2}$$

Figure 3 illustrates the qualitative result when transforming an RGB image into the representation space of our imager. The images labeled with (a) and (c) are captured respectively with an RGB image sensor and with our sensor, while figure (b) is obtained by applying the transformation of equations (1)-(2) on the RGB image.

A VGG-like [22] BNN model is trained either with the original RGB data of the CIFAR dataset and with binarized data obtained from the transformation (1)-(2) of the original data space. The model is composed of 12 convolutional layers and 2 fully-connected layers and is trained following the approach of [5]. Table 1 lists the accuracy on the test set composed of 10k samples picked from the CIFAR-10 dataset. The evaluation also includes a baseline floating point CNN model with the same VGG topology, trained with RGB and binarized data. The results show that the specific kind of imager binarization leads to an 18.8% performance drop with respect to the baseline, while the additional binarization of the model leads to a further 3.5% reduction. Despite the not-negligible accuracy degradation, the training process actually leads to model convergence and hence can be exploited for training event-based binarized neural networks for an application-specific scenario. We demonstrate in Section 5 that the accuracy degradation significantly attenuates by considering a reduced-complexity classification scenario, which is a more typical use-case for an always-ON sensing front-end.

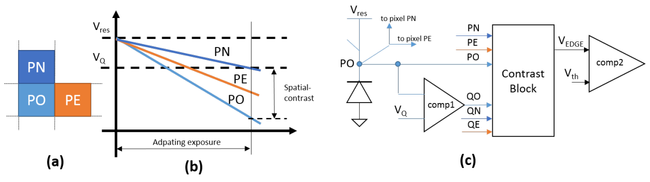

Figure 2: (a) In-sensor pixel mask to compute the binary gradient (b) Gradient extraction approach (c) Mixed-signal circuit for contrast extraction that is placed at every pixel location.

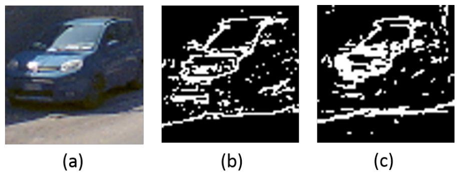

Figure 3: Image of a car taken with a RGB sensor (a) and with our imager (c). Image (b) is obtained by applying the transformation of eq.1-2 on the left RGB image to simulate the in-sensor processing.

Table 1: Accuracy on CIFAR10 dataset

Model	Accuracy
CNN with RGB input	91.36%
BNN with RGB input	86.78%
CNN with binarized input	72.50%
Event-Based BNN	68.94%

4 BNN ON A PARALLEL EMBEDDED PROCESSOR

In this section, we describe our software implementation of a Binarized Neural Network (BNN) on a parallel 4-core processor.

4.1 BNN Model

The BNN model is structured as a sequence of convolutions and fully-connected layers, where synaptic weights and neuron values are stored with 1-bit precision. When looking at the inference task, the convolution operation is expressed as:

$$\varphi(x, y) = \sum_{d,i,j} w(d, i, j) * I(d, i, j, x, y) \tag{3}$$

where $\varphi(x, y)$ is the output neuron's value at location (x,y) of one of the output channel, w and I are the weights and inputs banks, d is the input channel index and i, j are the spatial dimension indexes within the convolution filter. Because of $I, w \in \{0, 1\}$, equation (3) can be reduced to:

$$\varphi(x, y) = \text{popcount}(\mathbf{w} \text{ xnor } \mathbf{I}) \tag{4}$$

where $\mathbf{w}, \mathbf{I}$ are binary arrays that store the binary filter weights and inputs and popcount$(\cdot)$ returns the numbers of asserted bits of the argument. Note that the convolution output $\varphi(x, y)$ is an integer value. As presented by [5], the output is binarized after a batch normalization layer. We refer to the deterministic binarization that is defined as:

$$o_{bin}(x, y) = \text{sign}(\frac{\varphi(x, y) + b - \mu}{\sigma} \cdot \gamma + \beta) \tag{5}$$

where b is the convolution bias and μ, γ, σ and β are parameters learned by the batch normalization layer. These parameters are floating point but thanks to the $sign(\cdot)$ and the integer input $\varphi(x, y)$ the expression can be reduced to:

$$o_{bin}(x, y) = \begin{cases} \varphi(x, y) \geq \lfloor \mu - b - \beta \cdot \sigma/\gamma \rfloor & \text{if } \gamma > 0 \\ \varphi(x, y) \leq \lceil \mu - b - \beta \cdot \sigma/\gamma \rceil & \text{if } \gamma < 0 \end{cases} \tag{6}$$

therefore only an integer threshold and a single bit sign(γ) have to be stored for binarizing any output layer.

As shown by equations (4) and (6), a binarized convolution operation is then expressed as a couple of instructions, requiring bit-wise and popcount operations, along with an integer comparison.

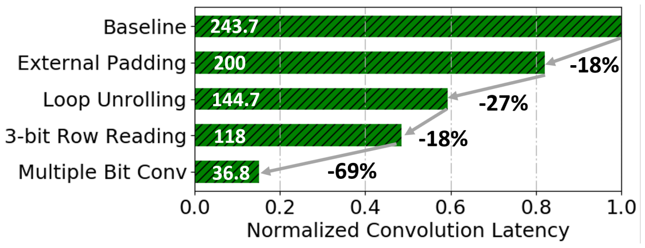

Figure 4: Average execution time to run a 3x3 convolution kernel for several optimization steps and their relative gains. The average number of clock cycles is reported in white on the bars.

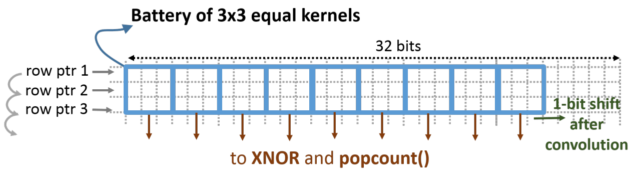

Figure 5: Optimized binary convolution kernel applied on every image plane. Three binary image rows are loaded and aligned with a battery of 3x3 convolution kernels. The masked popcount of the xored inputs resulting from any kernel position is accumulated and then the input is shifted to account all the possible input-filter alignment.

4.2 Software implementation and Memory Management

Our software implementation is based on the code architecture presented in [15]. Several optimizations have been performed on the original code version to accelerate the computation and, at the same time, minimize the memory footprint. As major improvements, (a) the binary weights needed to produce an output layer are stored contiguously in memory to minimize the memory requirement, (b) the XNOR operation is performed on 32-bit data registers, to fully exploit the datapath length and (C) thanks to the convolution formulation expressed by eq.(6), our implementation does not require any floating-point operation, with the exception of the last fully-connected layer, whose workload is however negligible in reference to the rest of the network inference task.

An ultra-low power parallel platform is employed for running the BNN inference [20]. The processor features a 4-core cluster with an extended OpenRISC instruction set. The cluster includes a 32KB tightly-coupled scratchpad memory, acting as L1 memory, while a 64KB L2 memory is placed on the off-cluster region. The architecture is implemented in 28nm FD-SOI technology. Power consumption measurement reported in the paper was measured on the silicon prototype of the SoC.

The BNN weight parameters are permanently stored in the L2 memory, besides the code region and the data input memory space. L1 memory serves for temporary storing of input and output data from the network layers. To this aim, two memory regions are sized as the maximum layer output capacity in the network, which is 2KB in case of the BNN topology of Tab. 2. At run-time, each core transfers a weight bank needed to produce a single output layer to a private L1 memory buffer, which has a size of 72 Bytes in this case.

4.3 Code Optimization and Parallelization

To optimize the BNN computation, we measure the execution time of the BNN model described in Tab. 2 by running the code on the instruction-accurate simulator of the parallel platform. Note that, in the proposed work, the BNN input is also binary, hence requiring only 1KB for transfer and storage, as opposed to the 12KB required by the system based on a RGB camera. When running on a single core, the baseline implementation spends the 96% of the computation on the convolution kernels of eq.(4), suggesting that this is the part that can benefit more from intensive optimization. For any output pixels, the mentioned operation consists of loading dx3x3 binary input pixels and compress them in 32-bit registers, to be xnored with the correspondent weights before the popcount. Fig. 4 illustrates the performance gain achieved by performing the optimization steps described below. The plot reports the average time to perform a 3x3 convolution, normalized with respect to the baseline. A first 18% cycles reduction is reached by padding every input layer before the convolution, hence avoiding to check border conditions in the inner loop. As a side effect, the L1 storage buffer has to be re-sized to contain the padded input (+13% memory for our network topology). An additional 27% reduction is achieved by exploiting loop unrolling. Doing this implies coding separately convolutional layers with different kernel size. For VGG-like models, which makes use only of 3x3 kernels, this is however not an issue. Others topologies featuring layers with different kernel sizes will pay an increased memory code footprint.

Going further, the implicit bit-level parallelism of data can be exploited by reading multiple input bits with a single load instruction. Thanks to this approach, any convolution of equation (4) requires only dx3 readings, because a 3-bit row can be loaded by using a single load operation. This allows to save an additional 18% of the computation time. Moreover, to fully exploit bit-level parallelism, the convolution operator can be applied to separate image input channels. The popcount results are accumulated along all the input channels before the binarization. Each image plane is tiled and scanned along the vertical direction. We exploit the spatial data order to load and analyze in parallel 32 binary pixels that belong to the same row. A battery of 9 identical binary filters is aligned to the loaded rows and xored. Figure 5 illustrates the process. For any of the filter masks, the popcount result is accumulated and the input is shifted. After this, the row pointers are increased by 1 up to reach the bottom line. The process repeats on the next vertical tiles for every input channel. A significant 69% cycles reduction is achieved thanks to this strategy. From a memory viewpoint, an accumulation L1 memory space is required, sized as the maximum input spatial layer dimension (64x64 in this case). Finally, we parallelize the BNN over the 4-cores by dividing the workload along the output feature dimension. This contributes to speed-up the code execution by 3.88x, which is close to theoretical maximum 4x.

5 EXPERIMENTAL RESULTS

5.1 Classification accuracy

The Event-Based BNN is evaluated with a real-life dataset, tailoring an always-on monitoring application for visual systems. A dual-camera setup, which includes a commercial RGB camera and our imager, is used for collecting 64x64 images, each one belonging to one of the three following categories: cars, cyclists and pedestrians.

Table 2: VGG-like BNN Model

Model Topology [1]
Conv3x3(#ich, 16) + MaxPool2x2
Conv3x3(16, 32) + MaxPool2x2
Conv3x3(32, 48) + MaxPool2x2
Conv3x3(48, 64) + MaxPool2x2
Conv3x3(64, 96) + MaxPool2x2
FC(384, 64)
FC(64,3)

The acquisition system synchronizes the data capture of the two sensors, which are physically aligned to match their fields of view. We collected two datasets for testing purpose, one with RGB images and one with binary gradient images, containing 100 samples per class each (see Fig. 2 for a sample image).

A binarized VGG-like model is trained to classify the acquired images. The network is composed of 5 convolutional layers and 2 fully-connected layers (for a total of 23Mop/img). Tab. 2 reports the network parameters layer by layer. A pooling layer is placed after every convolution. The presented topology is defined to fit the memory requirements of our architecture. In total, the memory footprint required by this model is less than 20KB. As a baseline for our event-based BNN, a binarized neural network with the same topology is trained and tested on 8-bit RGB data.

Both the BNN models are trained using Torch [4] for 100 epochs using the adaMax shift-based version proposed by [5] and a batch size of 128. Learning rate is set to 0.01 and divided by 10 every 15 epochs. To increase the generalization of the training process, the training and validation datasets are built by combining labeled patches from the KITTI [7] and MIO-TCD [1] datasets. Training data is augmented with random rotation to increase the number of training samples up to about 60k. The validation set is composed of 900 unique samples. When training the event-based BNN, training and validation data are binarized with equations (1)-(2). Trained models with the highest accuracy on the validation dataset report an accuracy on the real-life testing data of 84.6% and 81.6%, for RGB and gradient binarized input scenario, respectively. Therefore, the event-based BNN architecture presents a contained performance drop of 3% over the 3-classes application scenario.

5.2 Energy Evaluation

To assess the energy efficiency of the event-based BNN, we evaluate the energy consumption of the proposed visual node, which includes a smart image sensor [8] and a parallel processing unit running the BNN classifier (as described in Section 4). The proposed event-based binary visual node is benchmarked against a baseline system featuring a state-of-the-art low-power RGB imager [2] and the same processing unit, which runs a BNN with 8-bit 3 channels data inputs.

Table 3 reports the energy comparison between the event-based BNN and the baseline system for image acquisition and classification. It includes the contribution of the imager, sensor-to-processor data transfer and the 4-core processor that runs the binarized network. The active power of the processing unit is measured as 8.9mW when operating at 168MHz with a voltage supply of 0.7V.

[1]Conv3x3(x,y) is a convolutional layer with 3x3 weight filter size, x input layers and y output layers, FC(x, y) is a fully connected layer with x input neurons and y output neurons, #ich=3 for RGB input and #ich=1 for binary input

Table 3: Event Based BNN Energy Evaluation

Scenario	BNN with RGB input	Event-based BNN
Image Sensor Power Consumption	1.1mW @30fps	100μW @50fps
Image Size	324×244 (617kbits) [2]	128×64 (8kbits)
Image Sensor Energy for frame capture	66.7 μJ	2 μJ
Transfer Time (4bit SPI @50MHz)	3.1 msec	0.04 msec
Transfer Energy (8.9mW @0.7V)	28 μJ	2 μJ
BNN Execution Time (168MHz)	81.3 msec	75.3 msec
BNN Energy consumption (8.9mW @0.7V)	725 μJ	671 μJ
Total System Energy for Classification	820 μJ	674 μJ

Table 4: Event-Based vs Frame-based

Statistics per frame	Frame-Based	Event-based
Idle (no motion)		
Sensor Power	1.1mW (grey)	20μW
Avg Sensor Data	19764 Bytes	-
Transfer Time	790μsec	-
Processing Time	3.02 msec	-
Avg Processor Power	1.45mW	0.3mW (sleep)
Detection		
Sensor Power	1.1mW (grey)	60μW
Avg Sensor Data	19764 Bytes	~536 Bytes
Transfer Time	790μsec	21.4μsec
Processing Time	3.47 msec	187.6μsec
Avg Processor Power	1.57mW	0.511mW
Classification		
Sensor Power	2mW (RGB)	60μW
Avg Sensor Data	79056 Bytes	1024 Bytes
Transfer Time	3.16 msec	41μsec
Processing Time	81.3 msec	75.3 msec
Processor Energy	760 μJ	677 μJ

The platform is kept in active state during data transfer and BNN computation. Because of the higher amount of input data (a total of 324×244 8-bit pixels [2] instead of a single 128x64 binary channel [8]), the baseline scenario features a 77.2x higher data transfer time. Furthermore, the event-based BNN shows a smaller BNN computation time by 7.4% thanks again to the reduced amount of input data. Given all these contributions, the event-based BNN scenario reports a system-level energy reduction of 17.8% with respect to the baseline.

5.3 Continuous Sensing Evaluation

When dealing with always-on monitoring applications, we leverage the event based sensing paradigm to trigger the classification algorithm upon the detection of relevant events. On the contrary, when the sensor detects only a low number of events because of the reduced context activity, the system is kept in a low-power state. As a case-study of an always-on application, we focus on a parking entrance monitoring scenario, where an alert signal is triggered when a moving object (a car) enters the parking gate. To this aim, a detection and tracking algorithm is applied on the detected events to track the moving cars. An alert is generated whenever the object crosses a virtual gate corresponding to the parking entrance. This event triggers the execution of the binarized neural network used for object classification.

In the analyzed case-study, the sensing activity is driven by moving cars passing in front of the camera. Any of them generates 2.7 seconds of data recording. Tab. 4 reports the average statistics to acquire and elaborate the signals on both the event-based and the frame-based scenario, along with the different energy costs for

[2]The imager features a 324×244 resolution with Bayer color filter map, which roughly corresponds to a 3-channel QQVGA resolution

any of the following phases: *idle*, when no motion is detected by the mixed-signal image processing because of a static background, *detection*, aiming at generating alert signals, and *classification* upon detection, which implies transferring data and running the BNN classifier. Concerning the *idle* and *detection* phases, the energy cost is expressed in terms of average power consumption. We consider a frame rate of 30fps and the usage of duty cycling to reduce the energy consumption of the digital processor. When in sleep state, the processing unit consumes 0.3mW @0.7V, due to the memory region that cannot be power-gated because of data-retention. The energy contribution for the classification is derived from the analysis conducted in Section 5.2.

The event-based system keeps the digital platform in sleep state during the idle phase, due to the low amount of changed binary gradient pixels. Within the detection phase, a higher sensor datarate causes the processor to wake up for data processing. The power costs are still contained because of the limited number of pixels to be transferred and elaborated. Once a relevant event is detected through event-driven processing [14], the classification task is triggered. On the contrary, within the frame-based scenario, the sensor always transfers image data to the processor to determine the presence of relevant objects either on the idle or the detection phase. Hence, these phases are characterized by a similar power cost. The data analytic flow includes the background subtraction, morphological filtering and the extraction of the connected components. During the detection phase, the workload slightly increases due to the additional Kalman filtering and triggering process. As data source, the sensor provides QQVGA greyscale 8-bit data at a power cost of 1.1mW [2].

Fig. 6 shows the average power consumption for a varying number of moving objects per hour accessing to the parking gate. On the frame-based system the power is weakly dependent on the event frequency, while the event based system presents an increasing power cost due to the increased activation rate of the digital processor at a higher event rate. When the number of event per hour increases, the event-based visual system presents an energy saving of 4x, which can raise up to 8x at a reduced event activity.

6 CONCLUSION

The paper presented a combination of event-based sensing and binarized neural network to reduce the energy consumption of smart visual nodes. We demonstrated that combining an ultra low power imager with in-sensor hardwired capabilities and a software-programmable parallel processor reduces the system energy of 17.8% if compared to a baseline systems running a BNN classier, while paying only a 3% performance drop. Moreover, if considering a long-term monitoring application, the energy saving raises up to 8x thanks to the opportunistic event-based computing paradigm and the power-optimized hardware-software co-design of the system.

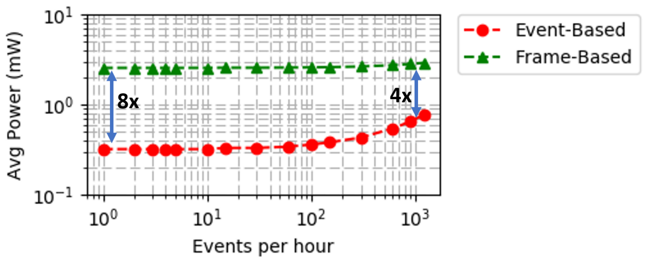

Figure 6: Average power consumption of event-based (in red) and frame-based (green) visual nodes with respect to a varying external event rate within an always-on monitoring application.

ACKNOWLEDGMENTS

This project was supported in part by the European project EuroCPS (g.a. 644090) and by the EU FP7 ERC Advanced project MULTITHERMAN (g.a. 291125).

REFERENCES

[1] http://podoce.dinf.usherbrooke.ca. The Traffic Surveillance Workshop and Challenge 2017 (TSWC- 2017).
[2] http://www.himax.com.tw/. (http://www.himax.com.tw/).
[3] R. Andri, L. Cavigelli, D. Rossi, and L. Benini. 2016. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights. In *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on.* IEEE, 236–241.
[4] R. Collobert, S. Bengio, and J. Mariéthoz. 2002. *Torch: a modular machine learning software library.* Technical Report. Idiap.
[5] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).
[6] S. K Esser, P. A Merolla, J. Arthur, A. Cassidy, R. Appuswamy, A. Andreopoulos, D. J Berg, Jeffrey L McKinstry, Timothy Melano, Davis R Barch, et al. 2016. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc of the National Academy of Sciences* (2016), 201604850.
[7] A. Geiger, P. Lenz, and R. Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conf. on Computer Vision and Pattern Recognition (CVPR).*
[8] M. Gottardi, N. Massari, and Syed A. Jawed. 2009. A 100μW 128x64 Pixels Contrast-Based Asynchronous Binary Vision Sensor for Sensor Networks Applications. *IEEE J. of Solid-State Circuits* 44, 5 (2009), 1582–1592.
[9] S. Jayasuriya, O. Gallo, J. Gu, and J. Kautz. 2016. Deep Learning with Energy-efficient Binary Gradient Cameras. *arXiv preprint arXiv:1612.00986* (2016).
[10] H. Kim, J. Sim, Y. Choi, and L. Kim. 2017. A Kernel Decomposition Architecture for Binary-weight Convolutional Neural Networks. In *Proc. of the 54th Annual Design Automation Conf. 2017.* ACM, 60.
[11] A. Krizhevsky and G. Hinton. 2009. Learning multiple layers of features from tiny images. (2009).
[12] E. Lee and S Wong. 2016. A 2.5 GHz 7.7 TOPS/W switched-capacitor matrix multiplier with co-designed local memory in 40nm. In *Solid-State Circuits Conf. (ISSCC), 2016 IEEE Int.* IEEE, 418–419.
[13] R. LiKamWa, Y. Hou, J. Gao, M. Polansky, and L. Zhong. 2016. RedEye: analog ConvNet image sensor architecture for continuous mobile vision. In *Proc. of the 43rd Intl. Symposium on Computer Architecture.* IEEE Press, 255–266.
[14] M Litzenberger, C Posch, D Bauer, AN Belbachir, P Schon, B Kohn, and H Garn. 2006. Embedded vision system for real-time object tracking using an asynchronous transient vision sensor. In *Digital Signal Processing Workshop, 12th-Signal Processing Education Workshop, 4th.* IEEE, 173–178.
[15] B. McDanel, S. Teerapittayanon, and H. Kung. 2017. Embedded Binarized Neural Networks. *Proc. of 2017 Int Conf Embedded Wireless Systems and Networks* (2017).
[16] P. Merolla, R. Appuswamy, J. Arthur, S. Esser, and D. Modha. 2016. Deep neural networks are robust to weight binarization and other non-linear distortions. *arXiv preprint arXiv:1606.01981* (2016).
[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conf. on Computer Vision.* Springer, 525–542.
[18] Á Rodríguez-Vázquez, R Carmona-Galán, J Fernández-Berni, V Brea, and JA Leñero-Bardallo. 2017. In the quest of vision-sensors-on-chip: Pre-processing sensors for data reduction. *Electronic Imaging* 2017, 11 (2017), 96–101.
[19] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, G. Tagliavini, A. Capotondi, P. Flatresse, and L. Benini. 2015. PULP: A parallel ultra low power platform for next generation IoT applications. In *2015 IEEE Hot Chips 27 Symposium (HCS).* 1–39.
[20] D. Rossi, I. Loi, A. Pullini, C. Mller, A. Burg, F. Conti, L. Benini, and P. Flatresse. 2017. A Self-Aware Architecture for PVT Compensation and Power Nap in Near Threshold Processors. *IEEE Design Test* 34, 6 (Dec 2017), 46–53.
[21] M. Rusci, D. Rossi, E. Farella, and L. Benini. 2017. A Sub-mW IoT-Endnode for Always-On Visual Monitoring and Smart Triggering. *IEEE Internet of Things J.* 4, 5 (Oct 2017), 1284–1295.
[22] K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[23] Y. Umuroglu, N. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers. 2017. Finn: A framework for fast, scalable binarized neural network inference. In *Proc. of 2017 Int. Symp. on Field-Programmable Gate Arrays.* ACM.