Stochastic Computing for Hardware Implementation of Binarized Neural Networks

Tifenn Hirtzlin, Bogdan Penkovsky, Marc Bocquet, Jacques-Olivier Klein, Jean-Michel Portal, Damien Querlioz

Date of publication 2019.

Digital Object Identifier TBA

Stochastic Computing for Hardware Implementation of Binarized Neural Networks

TIFENN HIRTZLIN[1], (Student, IEEE), BOGDAN PENKOVSKY [1], MARC BOCQUET [2], JACQUES-OLIVIER KLEIN [1] (Member, IEEE), JEAN-MICHEL PORTAL [2] and DAMIEN QUERLIOZ[1] (Member, IEEE)

[1]Centre de Nanosciences et de Nanotechnologies, Univ. Paris-Sud, CNRS, France
[2]Institut Matériaux Microélectronique Nanosciences de Provence, Univ. Aix-Marseille et Toulon, CNRS, France

Corresponding author: Tifenn Hirtzlin (email: tifenn.hirtzlin@c2n.upsaclay.fr), Damien Querlioz (email: damien.querlioz@c2n.upsaclay.fr)

This work was supported by the European Research Council Starting Grant NANOINFER (715872) and Agence Nationale de la Recherche grant NEURONIC (ANR-18-CE24-0009).

ABSTRACT

Binarized Neural Networks, a recently discovered class of neural networks with minimal memory requirements and no reliance on multiplication, are a fantastic opportunity for the realization of compact and energy efficient inference hardware. However, such neural networks are generally not entirely binarized: their first layer remains with fixed point input. In this work, we propose a stochastic computing version of Binarized Neural Networks, where the input is also binarized. Simulations on the example of the Fashion-MNIST and CIFAR-10 datasets show that such networks can approach the performance of conventional Binarized Neural Networks. We evidence that the training procedure should be adapted for use with stochastic computing. Finally, the ASIC implementation of our scheme is investigated, in a system that closely associates logic and memory, implemented by Spin Torque Magnetoresistive Random Access Memory. This analysis shows that the stochastic computing approach can allow considerable savings with regards to conventional Binarized Neural networks in terms of area (62% area reduction on the Fashion-MNIST task). It can also allow important savings in terms of energy consumption, if we accept reasonable reduction of accuracy: for example a factor 2.1 can be saved, with the cost of 1.4% in Fashion-MNIST test accuracy. These results highlight the high potential of Binarized Neural Networks for hardware implementation, and that adapting them to hardware constrains can provide important benefits.

INDEX TERMS Binarized Neural Network, Stochastic Computing, Embedded System, MRAM, In Memory Computing

I. INTRODUCTION

RECENT advances in deep learning have transformed the field of machine learning, with numerous achievements in image or speech recognition, machine translation and others. However, a considerable challenge of deep neural network remains their energy consumption, which limits their use within embedded systems [1]. The hardware implementation of deep neural networks is a widely investigated approach to increase their energy efficiency. A particularly exciting opportunity is to rely on in-memory or near-memory computing implementations [2]–[6], which are highly energy efficient as they avoid the von Neumann bottleneck entirely. This idea takes special meaning today, in particular with the emergence of novel memories such Resistive and Magnetoresistive Random Access Memories (RRAMs and MRAMs). Such memories are fast and compact non volatile memories, which can be embedded at the core of CMOS processes, and therefore provide an ideal technology for realizing in-memory neural networks [2], [3], [5].

A considerable challenge of this approach is that mod-

ern neural networks require important amounts of memory [7], which is not necessarily compatible with hardware in-memory computing approaches. Multiple roads have been explored to reduce the precision and memory requirements of neural networks. The quantization of the weights used for inference is the most natural route [8]. Architectural optimization can result in considerable reduction in terms of number of parameters and arithmetic operations, with only modest reduction in accuracy [9]. Network pruning [10] or network compression [11], [12] techniques, sometimes combining different methods, can allow implementing hardware neural networks with reduced memory access and therefore higher energy efficiency.

Binarized Neural Networks (BNNs) have recently appeared as one of the most extreme vision of low precision neural networks, as they go further than these approaches [13], [14]. In these simple deep neural networks, synaptic weights as well as neuron activations assume Boolean values. These models can nevertheless achieve state-of-the-art performance on image recognition, while being multiplier-less, and relying only on simple binary logic functions. First hardware implementations have already been investigated and have shown highly promising results [2], [6], [15], [16].

However, BNNs are not entirely binarized: the first layer input is usually coded as a fixed point real number. This fact is not a significant issue for operating BNNs on graphical processor units (GPUs) [13], as they feature extensive arithmetic units. Research aimed at implementing binarized neural network on Field Programmable Gate Arrays (FPGAs) [17] has also not specifically investigated the question of the non-binarized first layer: these works usually use the Digital Signal Processors (DSPs) of the FPGA to process the associated operations. However, in an application-specific integrated circuits (ASIC) implementation, the non-binarization of the first layer implies that this layer needs a specific design, which is more energy consuming and uses more area than the design used for the purely binary layers.

For this reason, in this work, we introduce a stochastic computing implementation of BNNs, which allows implementing them in an entirely binarized fashion. The network functions by presenting several stochastically binarized versions of the images to the BNN, in a way reminiscent to the historic concept of stochastic computing [18]. After presenting the background of the work (section II), the paper describes the following contributions.

- We show that this stochastic computing implementation of BNNs allows achieving high network performance in terms of recognition rate on the Fashion-MNIST and CIFAR-10 datasets. Stochastic BNN quickly approaches standard BNN performance when several stochastic binarized images are presented to the network. We also evidence that strategy for training stochastic computing BNNs should differ from the one used for conventional BNNs (section III).
- We design a full hardware ASIC in-memory BNN, which allows showing that the stochastic computing BNN strategy can save important area (62% on Fashion-MNIST) and energy (factor 2.1 on Fashion-MNIST with an accurat reduction of 1.4% with regards to a standard BNN (section IV). These numbers are discussed with regards to different alternative implementations.

II. BACKGROUND OF THE WORK

A. BINARIZED NEURAL NETWORKS

In this section, we first introduce the general principles of Binarized Neural Networks, an approach to considerably reduce the computation cost of inference in neural networks [13], [14]. In a conventional neural network with L layers, the activation values of the neurons of layer k, $a_i^{[k]}$, are obtained by applying a non-linear activation function f to the matrix product between real-valued synaptic weight matrix $W^{[k]}$ and the real-valued activations of the previous layer of neurons $a^{[k-1]}$:

$$a_i^{[k]} = f\left(\sum_j W_{ij}^{[k]} \cdot a_j^{[k-1]}\right). \tag{1}$$

In a BNN, excluding the first layer, neuron activation values as well as synaptic weights assume binary values, meaning $+1$ and -1. The products between weights and neuron activation values in Eq. (1) then simply become logic XNOR operation. The sum in Eq. (1) is replaced by the $\mathrm{popcount}$ operation, the basic function that counts the number of ones in a data vector. The resulting value is then converted to a binary value by comparing it to a trained threshold value $\mu_i^{[k]}$. Eq. (1) therefore becomes:

$$a_i^{[k]} = \mathrm{sign}\left(\underset{j}{\mathrm{popcount}}\left(\mathrm{XNOR}\left(W_{ij}^{[k]}, a_j^{[k-1]}\right)\right) - \mu_i^{[k]}\right), \tag{2}$$

where sign is the sign function.

Ordinarily, in binarized neural network, the first layer input X is not binarized. The implementation of operations for computing the first layer activations $a^{[1]}$ is therefore more complex than the basic XNOR and $\mathrm{popcount}$ operations:

$$a_i^{[1]} = \mathrm{sign}\left(\sum_j W_{ij}^{[1]} \cdot X_j - \mu_i^{[1]}\right). \tag{3}$$

Additionally, the thresholding operation is not performed on the last layer of the neural network. Instead, for the last layer, we identify the neuron with the maximum $\mathrm{popcount}$ value (i.e. the argmax of the last layer neurons), which gives the output of the neural network. The whole inference process of a conventional BNN is described with vectorized notations in Algorithm 1.

The performance of BNNs is quite impressive. A fully-connected BNN with two hidden layers of 1024 neurons, and the use of dropout during training [19] obtains a 1.8% error rate on the test dataset of the canonical MNIST handwritten digits task [20], with 300 epochs. In comparison, a

conventional neural network with no binarization and $\tanh$ activation function, and the same architecture and number of neurons, obtains a 1.5% test error rate after 300 epochs. Similarly, on more complex datasets such as CIFAR-10 or ImageNet, near-equivalent performed is obtained by BNNs and conventional neural networks [13], [14], [21]. The low memory requirements of BNNs (one bit by synapse), as well as the fact that they do not require any multiplication, makes them extremely adapted for inference hardware [2], [16], [22], [23].

The training process of BNNs is reminded in Appendix A. Unlike inference, the training process requires real valued weights and real arithmetic: training BNNs is not easier than in a conventional neural network. Therefore, a natural vision is to train BNNs on standard GPUs, and to use specialized ultra-efficient hardware only for inference.

Algorithm 1 Conventional BNN Inference Model

Require: input vector X, trained weight matrices W and threshold vectors μ

Ensure: predicted output

1. Non binary first layer:

$z^{[1]} \leftarrow W^{[1]} \cdot X$

$a^{[1]} \leftarrow \text{sign}(z^{[1]} - \mu^{[1]})$

2. Remaining layers:

for $k = 2 \rightarrow L$ **do**

$z^{[k]} \leftarrow \text{popcount}(\text{XNOR}(W^{[k]}, a^{[k-1]}))$

$a^{[k]} \leftarrow \text{sign}(z^{[k]} - \mu^{[k]})$

end for

$z^{[L]} \leftarrow \text{popcount}(\text{XNOR}(W^{[L]}, a^{[L-1]}))$

$output \leftarrow \text{argmax}(z^{[L]} - \mu^{[L]})$

In this work, we investigate how the first layer can be approximated by a stochastic input to decrease computing resources. This approach could also allow processing stochastic data for near sensor computing, which is a way to reduce considerably data transfer between sensors and data process. In addition, due to the possibility of implementing binarization from the first layer, the model can be completely generic with exactly the same architecture over the layers and allows reducing chip area.

B. STOCHASTIC COMPUTING

Stochastic computing is an approximate computing paradigm, known since the early days of computing [18], [24]. Nevertheless, hardware engineers have not exploited this computing scheme for processor design,as it requires applications that can be easily mapped with approximate computing. The principle is based on encoding all data as probabilities, represented as a temporal stochastic bitstreams: the number of ones among the bitstream represents the encoded probability. The main advantage of this encoding scheme is that mathematical functions can be easily approximated with simple logic gates. For instance a product is then implemented with a single AND gate, and a weighted adder can be implemented with a multiplexer gate [24]. Many arithmetic operations are therefore easy to implement with low power and small footprint characteristic. Despite these benefits, stochastic computing holds drawbacks: its limitation to low precision arithmetics, and the need to generate random bits. Random number generation can be a major part of the energy consumption, and, moreover, the generated random bits need to be uncorrelated.

Random bits have also found applications in the field of neural networks. The most widely used neural networks that intrinsically exploit stochasticity are the restricted Boltzmann machine, where each neuron is binary valued with a probability that depends on the previous layer neurons states [25]. An alternative technique to exploit stochasticity in neural networks is to approximate standard neural network architecture with stochastic computing. This approach as been proposed as early as the 1990's [26], and is currently being revisited in modern deep neural networks [27]–[29]. These works have shown promising results in terms of area and energy consumption. Typically, the largest challenge is the implementation of the non-linear activation function within the stochastic computing framework.

In this article, we suggest that stochastic computing is particularly adapted to the case of binarized neural network, as they work so naturally with bitstreams, and as the activation function is replaced by a simple thresholding operation.

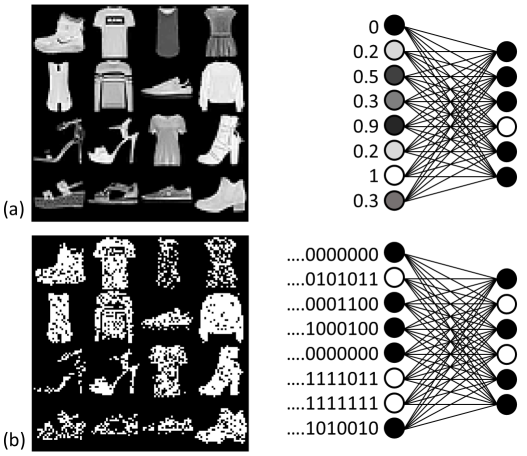

FIGURE 1: (a) In a conventional BNN, the first layer is not binarized. Grayscale input images are presented to the neural network. (b) In a stochastic computing-based BNN, binarized images are generated stochastically based on a grayscale image. Several binarized versions of the same original image can be presented sequentially to the neural network, following the basic principle of stochastic computing.

III. STOCHASTIC COMPUTING-BASED BINARIZED NEURAL NETWORK

To evaluate the stochastic computing approach, we use the Fashion-MNIST dataset, which has the same format as MNIST, but presents grayscale images of fashion items [30], and constitutes a harder task. The canonical MNIST dataset would not be appropriate for this study, as it con-

sists in images that are mostly black and white. As in the MNIST dataset, each image in Fashion-MNIST has 28x28 pixels, and can be classified within ten classes. The dataset contains $60,000$ training examples, $10,000$ test examples. Conventional BNNs (non-binarized first layer and no use of stochastic computing), perform very well on this task. With a fully connected BNN with first layer coded with eight bit fixed point real numbers, with two hidden layers of 1024 neurons each and dropout, a classification accuracy of 90% can be obtained after 300 epochs. This result is comparable with the test accuracy of 91% obtained by a conventional real-valued neural network with the same architecture.

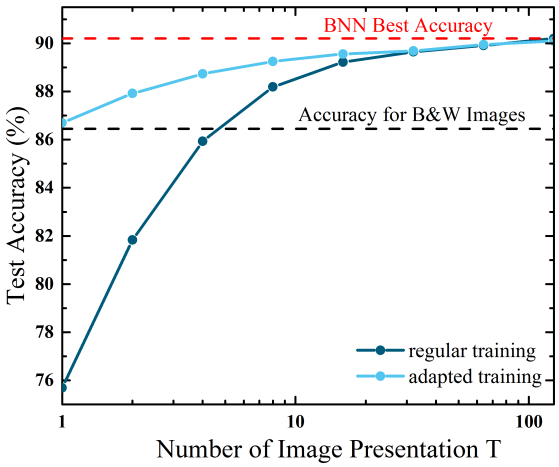

FIGURE 2: Accuracy on the Fashion MINIST classification task as function of the number of stochastic image presented for the two training methods. Navy blue curve: training of the neural network with grayscale images. Light blue curve: training with presentation of stochastic binarized images. Dashed black line: accuracy when training with a black and white image (i.e. pixels with a value greater than 0.5 are white and pixels that are smaller are black). Dashed red line: best accuracy when the binarized neural network is trained on Fashion-MNIST classification task with grayscale images. 300 training epochs were used.

A. STOCHASTIC COMPUTING WITH REGULAR TRAINING PROCEDURE

A first approach to design a stochastic computing BNN is to reuse the synaptic weights of a conventional BNN, trained with grayscale picture. However, in the inference phase, we approximate the computation of the first layer by using stochastic images presentation instead of grayscale images. The full inference algorithm is presented, in vectorized form, in Algorithm 2. An input X is transformed into binarized stochastic inputs X_t by taking the value of each grayscale pixel (between zero and one) as the probability for the corresponding pixel in the stochastic input to be one. Then, the networks computes $\text{popcount}(XNOR(W^{[1]}, X_t)) - \mu^{[1]}$, and sums the result of this computation over a number T of stochastic versions of the input X_t. Finally, the output of the layer is thresholded to obtain a binary value, and the rest of the neural network is computed in one pass in a fully binarized fashion.

Algorithm 2 Stochastic Computing BNN with Binarized First Layer

Require: X_t vectors: T stochastic binary versions of non-binary input X, trained weight matrices W and threshold vectors μ

Ensure: predicted output

1. Stochastic and binarized first layer

$z^{[1]} \leftarrow 0$

for $t = 1 \rightarrow T$ **do**

$z^{[1]} \leftarrow z^{[1]} + \text{popcount}(\text{XNOR}(W^{[1]}, X_t))$

end for

$a^{[1]} \leftarrow \text{sign}(z^{[1]} - T\mu^{[1]})$

2. Remaining layers

for $k = 2 \rightarrow L$ **do**

$z^{[k]} \leftarrow \text{popcount}(\text{XNOR}(W^{[k]}, a^{[k-1]}))$

$a^{[k]} \leftarrow \text{sign}(z^{[k]} - \mu^{[k]})$

end for

$z^{[L]} \leftarrow \text{popcount}(\text{XNOR}(W^{[L]}, a^{[L-1]}))$

$output \leftarrow \text{argmax}(z^{[L]} - \mu^{[L]})$

The quality of the results depends on the number of image presentation T. In Fig. 2, the navy blue curve shows the network test error as a function of T. We can see that after 100 stochastic image presentation, the accuracy is nearly equivalent to the use of grayscale images. With eight image presentation, the test accuracy is reduced to 88% instead of 90.1%. With a single presentation, the test accuracy is only 76%

B. ADAPTED TRAINING PROCEDURE

We now try a second strategy, where we train the neural network with binarized stochastic image presentation instead of grayscale images. To do this, during training, we use the conventional BNN training technique of Appendix A, but instead of using the normal grayscale Fashion-MNIST images, we use stochastic binarized ones, with the same number of presentation T as will be used during inference. The inference technique then remains identical to the one described in section III-A. In Fig. 2, in cyan color, we plotted the test error rate as a function of the number of presentation of the same image with this scheme. We see that the test accuracy is equivalent to the one obtained with grayscale images for high numbers of image presentation. On the other hand, with few stochastic presentation (one to five), the adapted input training technique allows reaching a quite high accuracy. If a single presentation is used at inference time, the network test accuracy is 86%. This test accuracy is equivalent to the one obtained when training a BNN with non-stochastic black and white versions of the Fashion-MNIST dataset (dashed black line in Fig. 2). If three image presentation are used, the network test accuracy increases to 88.7%.

These results show that when using the stochastic computing version of BNN, the adapted training procedure should be used.

C. CHOICE OF THE ACCUMULATION LAYER FOR STOCHASTIC SAMPLES

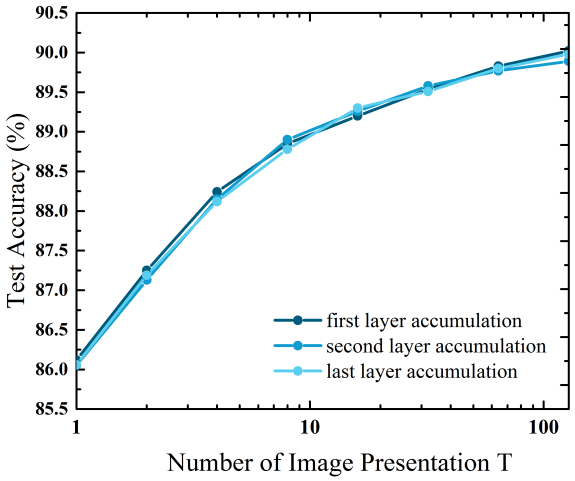

FIGURE 3: Accuracy on the Fashion MINIST classification task as function of the number of stochastic image presentation presented when the popcount value ias accumulated at different level of the network. The training was done with grayscale images and 300 training epochs were used.

Until now, at inference time, we have accumulated the outputs of the first layer over several presentations of the same image, then propagated the binarized output of the first layer to the other layers. An alternative strategy can be to perform the accumulation over the realizations of the input images at another layer. If the accumulation is done at the last layer, this procedure corresponds to using stochastic computing in the whole depth of the neural network.

Fig. 3 presents the test accuracy of the neural network on the Fashion-MNIST dataset, as a function of the number of presented realizations of the input images, for the different accumulation strategy, in networks trained with the adapted training strategy. This Figure shows that that the different accumulation strategy lead to equivalent accuracy, consistently with the principles of stochastic computing. The strategy of accumulation at the first layer is retained for the rest of the paper, as it allows for the minimum energy consumption.

D. EXTENSION TO THE CIFAR-10 DATASET

We now apply this strategy to the more advanced CIFAR-10 dataset. We use a convolutional neural network with six convolutional layers, with kernel size of three by three and a stride of one (number of filters 384, 384, 384, 768, 768 and 1536) and three fully connected layers (number of neurons 1024, 1024 and 10). Training is done in the same conditions as the Fashion-MNIST case, using dropout and Adam optimizer, and the pytorch deep learning framework. In the stochastic computing BNN, CIFAR-10 images are presented with binarized channel: each RGB channel pixel presents a value of zero or one. This value is chosen randomly with a probability equal to the RGB value of the corresponding

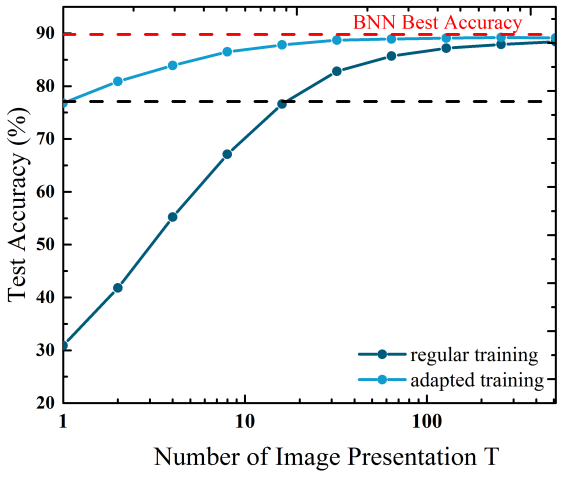

FIGURE 4: Accuracy on the CIFAR-10 classification task as function of the number of stochastic image presented for the two training methods. Navy blue curve: training of the neural network with color images. Light blue curve: training with presentation of stochastic binarized images. Dashed black line: accuracy when training with a binarized color image (i.e. RGB values with a value greater than 0.5 are white and pixels that are smaller are black). Dashed red line: best accuracy when the binarized neural network is trained on CIFAR-10 classification task with full color images. 2000 training epochs were used.

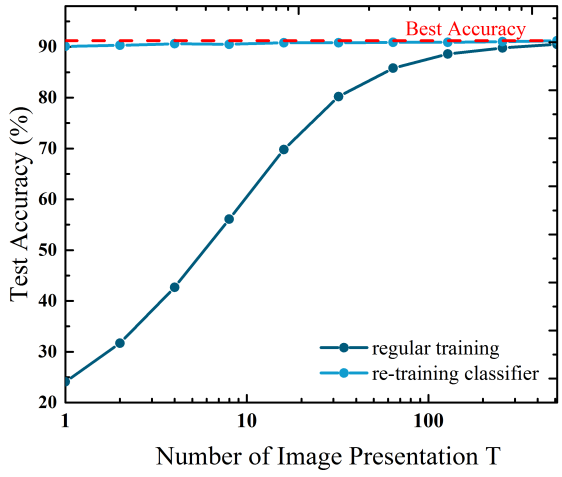

FIGURE 5: Accuracy on the CIFAR-10 classification task, but the stochastic computing approach is implemented at the end of the convolutional layers. Navy blue curve: training of the neural network in a conventional fashion. Light blue curve: classifier part of the neural network retrained with stochastic versions of the output of the convolutional layers. Dashed red line: best accuracy when the binarized neural network is trained on CIFAR-10 classification task with full color images. 2000 training epochs were used.

pixel of the image. Accumulation of stochastic realization is realized at the first layer, as described in section III-C.

Fig. 4 shows that the results on CIFAR-10 are very similar to the ones on Fashion-MNIST (Fig. 2). It present results obtained using the weights trained with full color images, and weights obtained with the adapted training approach. In both cases, the stochastic BNN results approach regular BNN results when the number of presentation T of stochastic

images is increased. The adapted training nevertheless gives highly superior results and should be preferred. This highlights that the stochastic BNN approach can be applicable to more complicated tasks than Fashion-MNIST.

We now consider a variation of this scheme, a partially binarized neural network. Fully connected layers of neural networks are particularly adapted for in-memory BNN implementation [2], [15], as these layers involve large quantities of memories. Convolutional layers are less memory intensive, and thus benefit less from binarization, while requiring increasing the number of channels when binarized [13]. In a hardware implementation, it can therefore be attractive to binarize only the classifier (fully connected) layers. In that case, the input of the classifier is real, and is processed with the stochastic BNN approach. This is also of special interest as the first fully connected layer in a convolutional neural network is usually the layer that involves the highest number of additions, and can therefore benefit significantly in a hardware to be implemented with the stochastic approach.

We consider a neural network with the same architecture as the fully binarized one, a reduced number of filters (128, 128, 128, 256, 256 and 512) and the same number of neurons in the fully connected layers (1024, 1024 and 10). Without the stochastic approach, this neural network has the same CIFAR-10 recognition rate than the fully binarized one (90%). Fig. 5 shows the results of the stochastic BNN with this approach. If the same weights are used than in a non stochastic BNN, the results look similar to the fully binarized approach of Fig. 4. On the other hand, if the classifier weights are retrained with the stochastic binarized inputs to the classifier, the stochastic results are very impressive. Even with a single image presentation T, the network approaches the performance of the non stochastic network. The stochastic BNN approach therefore appears especially effective in this situation.

IV. HARDWARE IMPLEMENTATION OF STOCHASTIC COMPUTING-BASED BINARIZED NEURAL NETWORK

In order to investigate the potential of the stochastic BNN approach, we designed a digital ASIC version of it using standard integrated circuit design tools. The architecture, presented in Fig. 6, allows performing the inference of a fully connected binary neural network of any size (up to 1024 neurons for each layer). The only parameter constrained by the hardware design is the number of weights that can be stored.

A. DESIGN OF THE ARCHITECTURE

Our architecture is inspired by the works of [31], with Static RAMs replaced by Spin Torque MRAM [32], and adaptation to stochastic computing. This architecture aims at performing inference on binarized neuronal networks with minimal energy consumption. To achieve this goal, it brings memory and computation as close as possible, to limit energy consumption related to data transfer. Such an architecture takes special interest with the emergence of new non-volatile memory components such as Spin Torque MRAM, which can be integrated within the CMOS manufacturing process, and which we consider here.

The architecture is described in detail in Appendix B, and can compute following a parallel or a sequential structure. The full design is made by a basic cell repeated 32x32 times (Fig. 6 (b-c)) that can perform both sequential or parallel calculation. It includes a 2 kbits memory array to stores weights, as well as XNOR gates and popcount logic.

We designed this system using the design kit of a commercial 28 nanometer technology. Digital circuits were described in synthesizable SystemVerilog description. MRAM memory arrays are modeled in a behavioral fashion, and their characteristics (area, energy consumption) are inspired by [33]. The system was synthesized to estimate its area and energy consumption. For energy consumption, we employed Value Change Dumps extracted from a Fashion-MNIST inference task, and estimated it using the Cadence Encounter tool.

B. ENERGY CONSUMPTION AND AREA RESULTS

Fig. 7(a) shows the area of a basic cell of our architecture (Fig. 6(b-c)), in the case of binary input (one operating bit), and in situations where the input is coded in Fixed Point representation (two, four and eight operating bit), as is required in the first layer of a conventional BNN. This Figure separates the area used by registers, logic and MRAM. A cell with binary input uses six times less area than a cell designed for eight bit input. Interestingly, the difference is mostly due to the $\mathrm{popcount}$ circuits, which need more depth when the input is non-binary. Similarly, as seen in Fig. 7(b), a cell with binary input uses 4.5 times less energy per cycle than the corresponding one with eight bits input. Again, the difference is mostly due to the $\mathrm{popcount}$ circuits.

The savings in terms of area transfer directly at the system level. We now consider the whole neural network used for Fashion-MNIST classification throughout section III. Using our architecture, a full BNN with eight bit first layer occupies $1.95\ mm^2$, while the BNN with stochastic binarized first layer occupies $0.73\ mm^2$, a 62% saving in area. These area values were extracted from a system designed for a T value of eight.

Fig. 8 plots the energy consumption for recognizing an image with our ASIC architecture, as a function of the number of presented stochastic images. This is compared with the energy cost of the same architecture, but using a non stochastic first layer, with eight bit input. We see that the system with stochastic first layer is more energy efficient than the system with non-binary first layer if less than eight presentation are used.

The previous curves do not include the cost of random bit generation. If we use a simple eight-bit Linear Feedback Shift Register (LFSR) pseudo random number generator, the added energy is $0.52nJ/cycle$, and the added area is $48,000\ \mu m^2$. Both are therefore negligible. It has also been shown that Spin Torque MRAM technology can be adapted to provide very low energy true random numbers [34]. If such a tech-

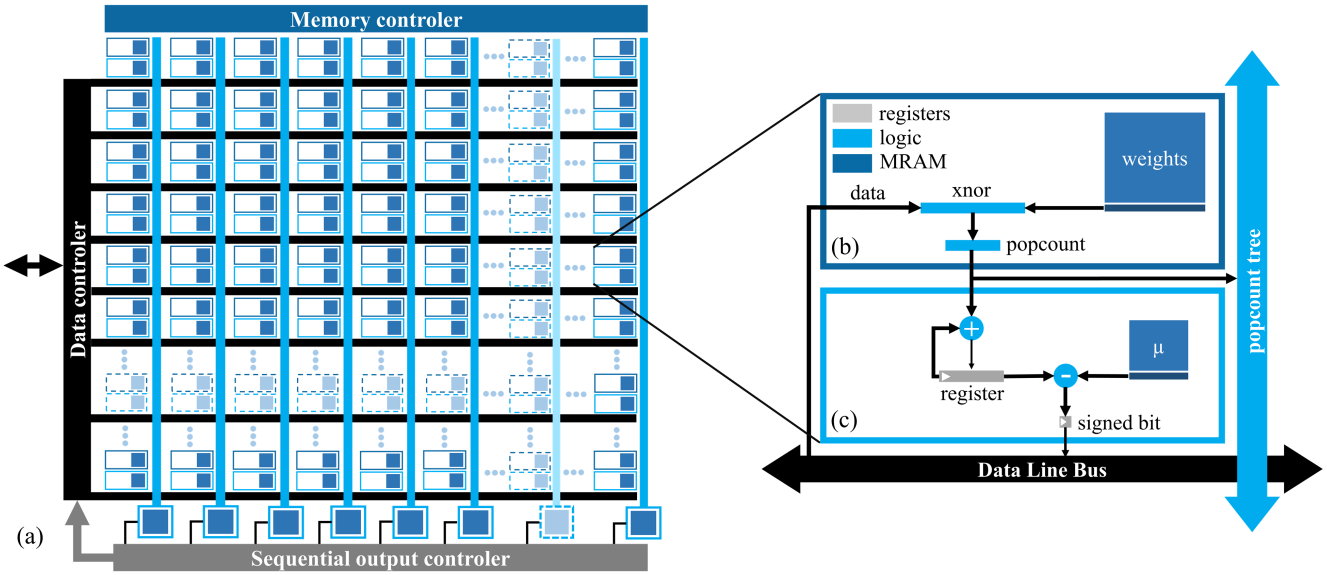

FIGURE 6: Design of an MRAM based fully connected binarized neural network, computing both parallel and serial. (a) Full architecture with 32×32 repeated cells. Each cell (b-c) behaves as a neuron if the input is sequential, or each column behaves as a neuron if the input is parallel.

nology was used, based on the numbers of [34], the energy cost of random bit generation would be $0.125nJ/cycle$, and the area much smaller than LFSR. The energy cost of random number generation is therefore negligible with regards to the consumption of the system seen in Fig. 8.

These energy numbers are very attractive with regards to non binarized implementations at equivalent recognition rate. Non binarized neural networks require less neurons and synapses than BNNs to achieve equivalent recognition rate. For example, to match the performance obtained in Fig. 2 on Fashion-MNIST with three image presentations ($T = 3$), one only needs a non-binarized neural network with eight-bit synapses with two layers of 500 neurons, while the BNN needs 1024 neurons per layer. However, in an ASIC, the non binarized neural network requires energy-hungry 8-bits multiplications and addition ($0.3\ pJ$ and $0.04\ pJ$ per operation in our $28\ nm$ technology). Taking into account only these arithmetic operations, the energy consumption is $220\ nJ$ for recognizing a Fashion-MNIST image with the same accuracy as the stochastic BNN with three image presentations. This stochastic BNN requires only $90\ nJ$ (Fig. 8), taking into account the whole system.

As a conclusion, this works highlights that the stochastic computing approach is attractive in terms of area occupancy. In terms of energy efficiency, it is very attractive if a relatively small number of presentation is used ($T < 8$). Therefore, it appears preferable to rely on the stochastic training approach seen in section III-B, and to use few stochastic image presentation for inference. For example, if three image presentation are used, a factor 2.1 can be saved on the energy consumption on Fashion-MNIST, with a reduction of 1.4% of test accuracy with regards to the best accuracy obtained by a BNN (dashed red line in Fig. 2). It should be noticed that the benefits of stochastic computing would be reduced on very deep neural networks, where the first layer plays a smaller role. Our approach is therefore the most promising for Internet-of-Things or sensor networks applications, where relatively small neural networks can provide sufficient intelligence, but circuit cost and energy consumption are the most critical issues. On deep neural networks, nevertheless, the approach of implementing only the classifier with a stochastic BNN, as mentioned in section III-D, can be of high interest.

V. CONCLUSION

In this work, we presented a stochastic computing approach to Binarized Neural Networks. This allows implementing them in an entirely binarized fashion, whereas in conventional BNNs, the first layer is not binary. We showed that the stochastic computing approach can reach recognition results similar to the conventional approach. We identified that for highest accuracy, the neural network should not be trained with regular images as conventional BNNs: it it is more beneficial to train stochastic BNNs with stochastic binarized images, using the same number of image presentation as will be used during inference. The design of a full BNN ASIC relying on in-memory computing, then highlighted the benefits of BNNs in terms of area and energy consumption. Stochastic BNNs allow using the same compact architecture for all layers, which leads to strong benefits in terms of area (62% reduction in the case of Fashion-MNIST classification). In terms of energy, the benefits can be very strong if we accept a slight reduction in classification accuracy. For example, on Fashion-MNIST classification, we can reduce the energy consumption by a factor 2.1, with a decrease of 1.4%

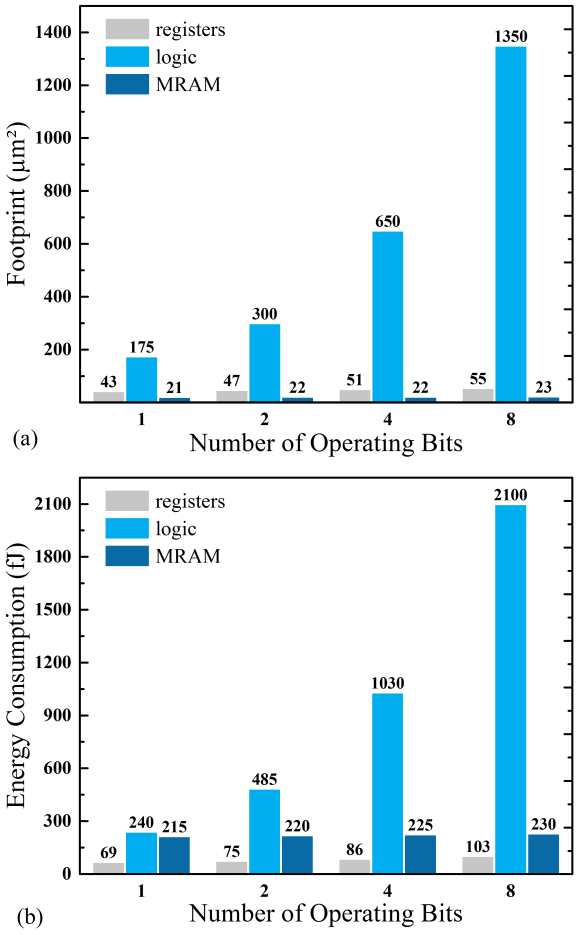

FIGURE 7: (a) Area of the basic cell (Fig. 6 (b-c)) of our ASIC architecture, implemented in a 28 nm CMOS technology, as function of the number of operating bit for a fixed point binary architecture. One-bit corresponds to our stochastic fully binarized architecture. (b) Corresponding energy consumption, per clock cycle.

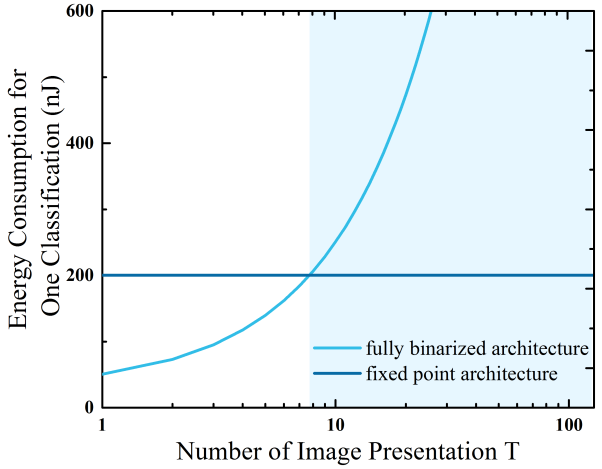

FIGURE 8: Energy consumption of the full Fashion-MNIST classifier systems, for the classification of one image. Light blue: stochastic fully binarized binary architecture. Navy blue: Conventional BNN with non binary (8 bit fixed point) first layers. The neural networks have two layers with 1024 neurons each. The light blue area indicate the regime where the non-binary first layer is more energy efficient thant the fully binarized system.

in classification accuracy.

These results highlight the high potential of BNNs for implementing compact and energy efficient in-memory neural networks, and the potential of stochastic approaches for hardware artificial intelligence. Future works should focus on the physical implementation of the proposed scheme, as well as the extension of the approach to other tasks than vision, such as medical tasks, where energy efficiency can be a particularly important concern.

.

APPENDIX A TRAINING ALGORITHM

Throughout the paper, neural networks are trained with the algorithm proposed by Courbariaux et al in [13]. This algorithm relies on two fundamental principles. First, the function $\mathrm{Clip}(x,-1,1)$ is used instead of the sign function in the backpropagation phase, as it can be differentiated. Second, the binarized weights W are not directly modified during the back propagation: their modification is done indirectly through the modification of the real weight W_a associated with each synapse.

Our design includes two modifications with regards to the work of [13]. In the original paper, the multi-layer perceptron trained on MNIST consisted of hidden layers of binarized units, topped by L2-SVM output layer. Here, we used a $\mathrm{softmax}$ output layer. Second, the parameters γ and β used for the batch normalization were not trained, and we used $\gamma = 1$ and $\beta = 0$ instead. The complete algorithm that we used is presented in Algorithm 3.

APPENDIX B DESCRIPTION OF THE ASIC BNN ARCHITECTURE

The architecture for hardware implementation of BNN inference is presented in Fig. 6. The basic function of a BNN is to compute $\mathrm{popcount}(\mathrm{XNOR}(W,X)) - \mu$. To perform this function, first, the system needs to perform the XNOR between the inputs X and the weights W, stored in the Spin Torque MRAM memory blocks. Second, it needs to perform the $\mathrm{popcount}$ function, and then compare this value with a threshold.

To achieve this goal, the architecture is made of basic cells (cell Fig. 6 (b-c)), composed of a 2 kbits memory array that store weights, 32 XNOR logic gates that perform the XNOR between the 32 bits weights and the 32 bits received data, a 32 bits to 5 bits popcount module compound of basic tree adders. The basic cell is repeated 32x32 times.

The architecture can be operated with a "parallel to sequential" structure, or a "sequential to parallel" structure. The sequential to parallel structure allows dealing with long input sequence data, and outputs a limited parallel output data. By contrast, the parallel to sequential structure allows dealing with limited parallel input data, and outputs long

Algorithm 3 Conventional BNN training model

Require: training data : X_{train}, targets output y_{train}, previous binarized and real weights W and W_a, and previous threshold values μ

Ensure: updated weights W_{t+1} and $W_{a,t+1}$, updated BatchNorm parameters μ and σ

1. Forward propagation

for $k = 1$ to L **do**

$W^{[k]} \leftarrow \text{sign}(W_a^{[k]})$

$z^{[k]} \leftarrow W^{[k]} \cdot a^{[k-1]}$

$\widehat{z}^{[k]} \leftarrow \text{BatchNorm}(z^{[k]}, \mu^{[k]}, \sigma^{[k]})$

if $(k < L)$ **then**

$a^{[k]} \leftarrow \text{sign}(\widehat{z}^{[k]})$

else

$a^{[k]} \leftarrow \text{softmax}(\widehat{z}^{[k]})$

end if

end for

Compute gradient of softmax cross entropy loss :

$$g_{a^{[L]}} = \frac{\partial C}{\partial a^{[L]}} = a^{[L]} - y$$

2. Backward propagation

for $k = L$ to 1 **do**

if $(k < L)$ **then**

$g_{a^{[k]}} \leftarrow g_{a^{[k-1]}} \circ 1_{|a_{k<1}|}$

end if

$g_{\widehat{z}^{[k]}} \leftarrow \text{BackBatchNorm}(g_{a^{[k]}}, \widehat{z}^{[k]}, \mu^{[k]}, \sigma^{[k]})$

$g_{z^{[k]}} \leftarrow {W^{[k]}}^T g_{\widehat{z}^{[k]}}$

$g_{W_b^{[k]}} \leftarrow g_{\widehat{z}^{[k]}}\ a_{k-1}^T$

end for

3. Update parameters

for $k = 1$ to L **do**

$W_{a,t+1}^{[k]} \leftarrow \text{Clip}(\text{UpdateAdam}(W_{a,t+1}^{[k]}, g_{W_b^{[k]}}), -1, 1)$

$(\mu^{[k]}, \sigma^{[k]})_{t+1} \leftarrow \text{MovingAverage}(\mu_B^{[k]}, \sigma_B^{[k]})_t$

end for

sequence data. The basic cells of Fig. 6 (b-c) can perform both, sequential or parallel calculation. The output of the popcount can be given to the sequential part of the cell or to the parallel part of the system that will perform the popcount through the whole column, with a "popcount tree" module shared with all the cells of the column. The sequential section of the cell that receive the popcount output will perform the full popcount operation sequentially by summing the popcount output using a register.

To perform the activation function of the neuron, the system adds in each cell the threshold values μ in a memory array. The signed bit of the difference between the popcount value saved in the register and μ gave the activation value. The same operation is made with the output of the popcount tree shared along the column.

REFERENCES

[1] Editorial, "Big data needs a hardware revolution," Nature, vol. 554, no. 7691, p. 145, Feb. 2018.

[2] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," Proc. IEEE, vol. 106, no. 2, pp. 260–285, 2018.

[3] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," Nature Electronics, vol. 1, no. 6, p. 333, 2018.

[4] D. Querlioz, O. Bichler, A. F. Vincent, and C. Gamrat, "Bioinspired programming of memory devices for implementing an inference engine," Proc. IEEE, vol. 103, no. 8, pp. 1398–1416, 2015.

[5] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola et al., "Neuromorphic computing using non-volatile memory," Advances in Physics: X, vol. 2, no. 1, pp. 89–124, 2017.

[6] E. Giacomin, T. Greenberg-Toledo, S. Kvatinsky, and P.-E. Gaillardon, "A robust digital rram-based convolutional block for low-power image processing and learning applications," IEEE Transactions on Circuits and Systems I: Regular Papers, 2018.

[7] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," arXiv preprint arXiv:1605.07678, 2016.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," The Journal of Machine Learning Research, vol. 18, no. 1, pp. 6869–6898, 2017.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[10] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). IEEE, 2016, pp. 267–278.

[11] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in International Conference on Machine Learning, 2015, pp. 2285–2294.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[13] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," arXiv preprint arXiv:1602.02830, 2016.

[14] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in Proc. ECCV. Springer, 2016, pp. 525–542.

[15] M. Bocquet, T. Hirztlin, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "In-memory and error-immune differential rram implementation of binarized deep neural networks," in 2018 IEEE International Electron Devices Meeting (IEDM). IEEE, 2018, pp. 20–6.

[16] E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh, and D. Marr, "Accelerating binarized neural networks: comparison of fpga, cpu, gpu, and asic," in Proc. FPT. IEEE, 2016, pp. 77–84.

[17] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable fpgas," in Proc. Int. Symp. FPGA. ACM, 2017, pp. 15–24.

[18] B. R. Gaines, "Stochastic computing systems," in Advances in information systems science. Springer, 1969, pp. 37–172.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in Advances in Neural Information Processing Systems, 2017, pp. 345–353.

[22] X. Sun, S. Yin, X. Peng, R. Liu, J.-s. Seo, and S. Yu, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," Proc. DATE, vol. 2, p. 3, 2018.

[23] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in Proc. ASP-DAC. IEEE, 2017, pp. 782–787.

[24] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," ACM TECS, vol. 12, no. 2s, p. 92, 2013.

[25] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural computation, vol. 18, no. 7, pp. 1527–1554, 2006.
[26] S. L. Bade and B. L. Hutchings, "Fpga-based stochastic neural networks-implementation," in Proc. IEEE Workshop on FPGAs for Custom Computing Machines. IEEE, 1994, pp. 189–198.
[27] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "Vlsi implementation of deep neural network using integral stochastic computing," IEEE Trans. VLSI Systems, vol. 25, no. 10, pp. 2688–2699, 2017.
[28] A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, J. Li, X. Qian, and B. Yuan, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," ACM SIGOPS Operating Systems Review, vol. 51, no. 2, pp. 405–418, 2017.
[29] V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rosselló, "A new stochastic computing methodology for efficient neural network implementation," Proc. IEEE TNNLS, vol. 27, no. 3, pp. 551–564, 2016.
[30] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, 2017.
[31] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, T. Kuroda et al., "Brein memory: A 13-layer 4.2 k neuron/0.8 m synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm cmos," in Proc. Symp. VLSI Circuits. IEEE, 2017, pp. C24–C25.
[32] D. Shum, D. Houssameddine, S. Woo, Y. You, J. Wong, K. Wong, C. Wang, K. Lee, K. Yamane, V. Naik et al., "Cmos-embedded stt-mram arrays in 2x nm nodes for gp-mcu applications," in Proc. Symp. VLSI Tech. IEEE, 2017, pp. T208–T209.
[33] K. C. Chun, H. Zhao, J. D. Harms, T.-H. Kim, J.-P. Wang, and C. H. Kim, "A scaling roadmap and performance evaluation of in-plane and perpendicular mtj based stt-mrams for high-density cache memory," IEEE JSSC, vol. 48, no. 2, pp. 598–610, 2013.
[34] D. Vodenicarevic, N. Locatelli, A. Mizrahi, J. S. Friedman, A. F. Vincent, M. Romera, A. Fukushima, K. Yakushiji, H. Kubota, S. Yuasa et al., "Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional computing," Physical Review Applied, vol. 8, no. 5, p. 054045, 2017.

TIFENN HIRTZLIN is a PhD student in Electrical Engineering at Université Paris-Sud. He received the M.S. degree in Nanosciences and Electronics from the University Paris-Sud, France, in 2017. His work focuses on designing intelligent memory-chip for low energy hardware data processing using bio-inspired concepts as probabilistic approach to brain function or more classical neural network approaches.

BOGDAN PENKOVSKY is a postdoctoral CNRS researcher at Paris-Sud University. He received his M.S. degree in Applied Mathematics from the National University of Kyiv-Mohyla Academy, Ukraine, in 2013 and the Ph.D. degree in optics and photonics applied to neuromorphic computing from the University of Burgundy - Franche-Comté, France, in 2017. His work is on intelligent, low energy hardware design for biomedical applications.

MARC BOCQUET is an Associate Professor in the Institute of Materials, Microelectronics and Nano-sciences of Provence, IM2NP at Univerisity of Aix-Marseille. He received the M.S. in electrical engineering degree in 2006 and the Ph.D. degree in electrical engineering in 2009, both from the University of Grenoble, France. His research interests include memory model, memory design, characterization and reliability.

JACQUES-OLIVIER KLEIN (M'90) received the Ph.D. degree from Univ. Paris-Sud, France, in 1995. He is currently Full Professor at Univ. Paris-Sud, where he focuses on the architecture of circuits and systems based on emerging nanodevices in the field of nanomagnetism and bio-inspired nanoelectronics. In addition, he is lecturer at the Institut Universitaire de Technologie (IUT) of Cachan. He is author of more than one hundred technical papers.

JEAN-MICHEL PORTAL (M'87) is a Full Professor in the Institute of Materials, Microelectronics and Nano-sciences of Provence, IM2NP at Univeristé of Aix-Marseille. He received the Ph.D. degree in 1999 from University of Montpellier 2, France. From 1999 to 2000, he was temporary researcher at University of Montpellier 2 in the field of FPGA design and test. From 2000 to 2008, he was assistant professor at the Univ. of Provence, Polytech Marseille, and conducted research activities in L2MP in the field of Memory testing and diagnosis, test structure design and design for manufacturing. In this position he participated to industrial project on non-volatile memory testing and diagnosis with ST Microelectronics. In 2008, he became Full Professor at Aix-Marseille Univ. and since 2009 he heads the "Memories Team" of the IM2NP. His research fields covers design for manufacturing and memory design, test and reliability.

DAMIEN QUERLIOZ (M'08) is a CNRS Research Scientist at Univeristé Paris-Sud. He received his predoctoral education at Ecole Normale Supérieure, Paris and his PhD from Université Paris-Sud in 2008. After postdoctoral appointments at Stanford University and CEA, he became a permanent researcher at the Centre for Nanoscience and Nanotechnology of Université Paris-Sud. He focuses on novel usages of emerging non-volatile memory, in particular relying on inspirations from biology and machine learning. Damien Querlioz coordinates the INTEGNANO interdisciplinary research group. In 2016, he was the recipient of an European Research Council Starting Grant to develop the concept of natively intelligent memory. In 2017, he received the CNRS Bronze medal.