Djohan Bonnet

Nanotech 2020

Cea Leti

Ex-situ transfer of deterministic and bayesian neural networks to resistive memory inference hardware

From 17/02/2020 to 14/08/2020

Confidentiality : yes

Under the supervision of :

- Elisa Vianello elisa.vianello@cea.fr
- Liliana Prejbeanu liliana.buda@cea.fr

Ecole nationale supérieure de physique, électronique, matériax

Phelma

Bât. Grenoble INP - Minatec

3 Parvis Louis Néel - CS 50257

F-38016 Grenoble Cedex 01

Tél +33 (0)4 56 52 91 00

Fax +33 (0)4 56 52 91 03

http://phelma.grenoble-inp.fr

Contents

Acknowledgements

The internship took place in CEA-Grenoble, inside the LETI (Laboratoire d'Electronique et des Technologies de l'Information) composed of 1900 researchers with 50 years history. The CEA-LETI work closely to industries (250 industrial agreement) in key sectors of technolgies. The LETI is again divided in departments. More precisely, I was working inside the memory component laboratory directed by Dr NOWAK Etienne belonging to the department DCOS ('Département COmposants Silicium') directed by Dr FAYNOT Olivier. I was supervised by Dr VIANELLO Elisa and I worked closely with two Phd students Mr DALGATY Thomas and Mr ESMANHOTTO Eduardo. I would like to thanks Dr VIANELLO Elisa, Eduardo ESMANHOTTO, Dr NOWAK Etienne for the teaching, help and welcom they provided me. And I would like to give a special thanks to Thomas DALGATY who helped and adviced me on a daily basis including during the lockdown. This internship gave me a Phd offer and the opportunity to participate to the writting of an article with the LETI team over deterministic neural network application that will be submitted the 31th of August to IEDM. Furthemore the writting of second article concerning bayesian neural network is in discussion.

CEA LETI

Grenoble, August 2020

1 Introduction

Neuromorphic computing circuits are one of the hot topics on the 21th century. The neuromorphic family subject to this study has a behavior very close to classical neural network used for deep learning. They are being subject to a lot of research since the past decade for their ability to apply the dot product operation in a energy, time and space efficient analogical manner [2, 3]. The main advantage of neuromorphic architecture is to save the journey of data between the memory and computing part of the system. Thus in-memory computing system often appears as a solution for embedded application that requires low power consumption and high integration density [4]. Those systems can be both trained ex-situ or in-situ. The in-situ trainning can provide large speed-energy efficiency for the training while adapting itself to hardware imperfections but raises more technical issues that are yet to be solved [5, 6]. The training ex-situ is usefull for at the edge computing as the training has to be done only once then can be transferred on all the inference hardware. Here resistive random-access memory (RRAM) technologies are a key component of the system, they are related to the synapses weight of the neural network. Their device to device, cycle to cycle variability and retention time as well as the implied weight discretization directly deteriorate the computation. An implementation principle where the neural neural synaptic weight and bias are embedded with a pair of conductance has been chosen [1]. Several experimental datas concerning RRAMs electrical characterisation were available in the CEA-LETI laboratory. From those starting points the internship consisted to simulate neural networks built with those RRAM technologies and this implementation strategy. However several choice concerning the neuromorphic circuit topology, type and also RRAM programming strategy are possible. The aim of the study is thus to design several neural network and modify them so that they simulate the performance they would have if they were implemented on an RRAM based inference hardware. By this mean, investigate the trade off between RRAM programming energy cost, circuit size and prediction accuracy depending on the application and its requirement. The Neurals Networks of this report are fully connected and used for the MNIST and heart beat disease classification task. For the second application, the question of prediction uncertainty is also asked. Firstly a description of the neural network used and the expiremental data at disposal is provided in order to understand the rest of the study.

Then the simulation strategy developped during this internship is described. Finally the results are presented and analyse in order to draw a conclusion and the perspectives of this experiment.

2 Background

2.1 Insights

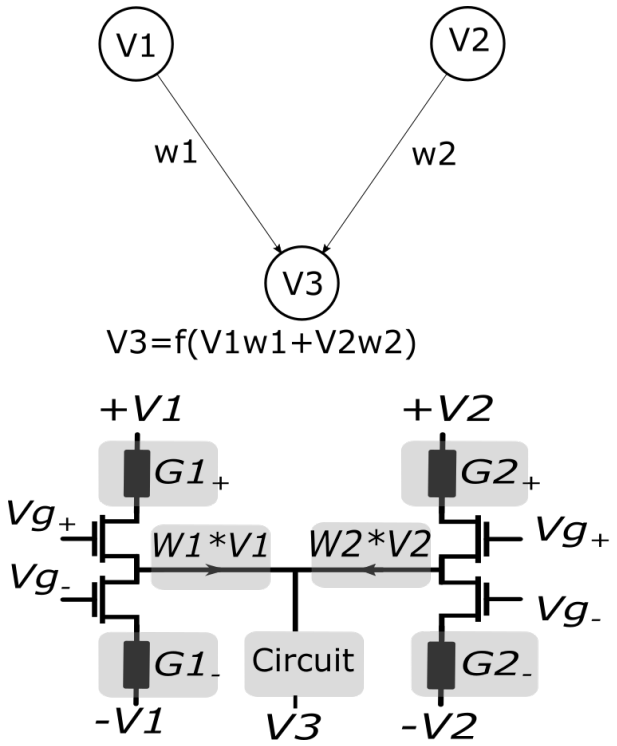


Figure 2.1: Working principle of weight implementation by conductance.The vertices Vi represent the neurons, their outputs are voltages, the weights Wi characterize the synapses that connect them. The output of $V3$ is define by both the activation function f and the dot product of the weight and previous output neurons vectors. G_+ and G_- are two resistives memories, $W = G_+$ - G_- represent the weight of the synapse. The current $V1.W1 + V2W2$ is the input of a specific circuit, the output of the whole system is a voltage $V3 = f(V.g)$.[1]

The whole part of this report has only the aim to present the working principle of the neural networks considered from the device to theoretical point of view. This presentation is needed to understand the way experimental data are used in the simulation and what the result means. Classical neural networks are made by neurons V_i interconnected by synapses with a certain weight w_i. The output of the neuron V_3 is determined by both what is called an activation function f and the dot product of the previous vector of neurons and the weights of their respective synapses. This dot product or multiply accumulate operation can be physically implemented by Kirchoff's laws where the weights are embodied by the conductance and neuron value by a voltage [7, 8, 9]. Since in software model the weights can be positive or negative, the quantity considered is the subtraction between a pair of

conductance following the idea presented in the paper [1]. Then a specific circuit will convert the current V.g into a voltage in a way that match the activation function desired. In this study the conductance used are RRAM devices, those non volatile memories are well suited for in memory application due to their scalability and compatibility with the CMOS process flow [8]. The RRAM working principle shown on figure 2.2 is based on filamentary conduction, an oxygen vacancies filament can allow a device to be in go from LCS to HCS [7]. One insteresting properpety is that the HCS can be tuned into separable state that allows a multi level programation [8]. However that last property is limitated by the stochasticity intrasic to RRAM and its finite conductance range. The figure 2.1 is made to summarize this subsection, and to get the idea of how the simplest neural network can be implemented. Now that the physical principles has been shown to build neural network in an analogical manner, the bottom up presentation strategy will continues to finish this background part. Since the deterministic neural network are more classical they will be presented at first.

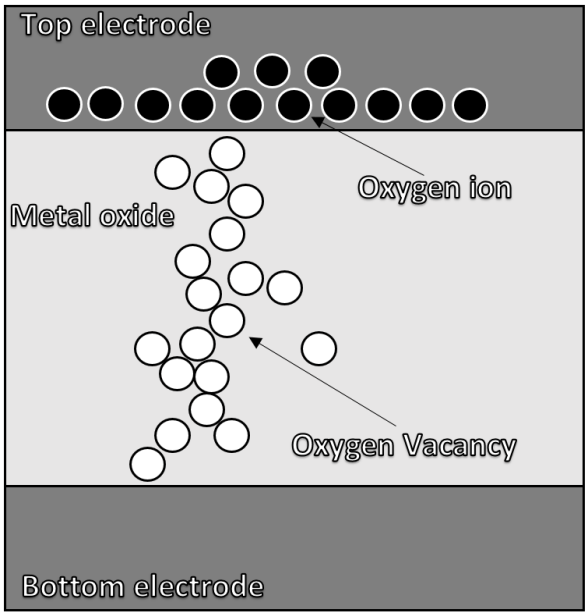


Figure 2.2: Physical explanation of filamentary resistive memories, the white dots represent the filament formed by oxigen vacancies that allows the resistance to be in HCS

2.2 Deterministic Neural Network

Deterministic neural network (DNN) are directly related to what was presented in the previous part. The weights of synapses and bias are represented by scalars (32 bit float

in the built $TensorFlow$ model). Several architecture are possible to connect neurons and synapses, the study here focused on the more basic one, the fully connected neural network. The easier way to understand the principle is to take an example. In the figure 2.3 the neural network architecture used for MNIST application is represented. In the MNIST database are represented handwritten digit on a 28x28 pixel image. The Input layer is thus composed of 784 inputs and the outputlayer of 10 outputs corresponding to the label. The output with the greater value represent the choice of the circuit, if $output_i$ is the greater outputs the circuit is predicting that the digit (i-1) is represented by the image. During the training of the DNN the two essential parts are the loss function and the optimizer. Several images with a known label are feed to the circuit, the loss function measures how far from the optimal output the circuit is. The optimal output being a 1 on the neuron corresponding to the label and 0 everywhere else. The the optimizer will adjust the value of all the synapses to minimizes the lossfunction using the gradient descent. The operation is done as much as needed to find a proper setting to the synapses. The loss function used in this study is Categorical cross entropy and the optimizer ADAM. ADAM optimiser is one of the most famous gradient based optimization and present the advantages of needing very little tunning of *hyperparameters* [10]. To conclude this part a deterministic neural network is a function with as much parameter as bias and synapses. The training consist to find the best function inside the space function. The space has a finite number of dimension, the next neural network does not present this particularity theoretically.

2.3 Bayesian Neural Network

Bayesian Neural Network (BNN) are a probabilistic interpretation of DNN. The synapses of the neural network became distribution instead of scalar (see figure 2.4). They have shown several advantages compared to DNN for over fitting issues [11] and gives more information concerning the uncertainty of the prediction [12]. Their name came from the famous mathematician due to the use of Bayesian probability theory. For the classical neural network given training datas x_i and their label y_i, the training consist of finding a function f define by the weight value of the synapses and bias of the network such that $f(x) = y$. Here the idea is a bit more sophisticated, first we define what is called a *prior* distribution $p(f)$. Which means that a prior belief of the distributions has to be chosen for

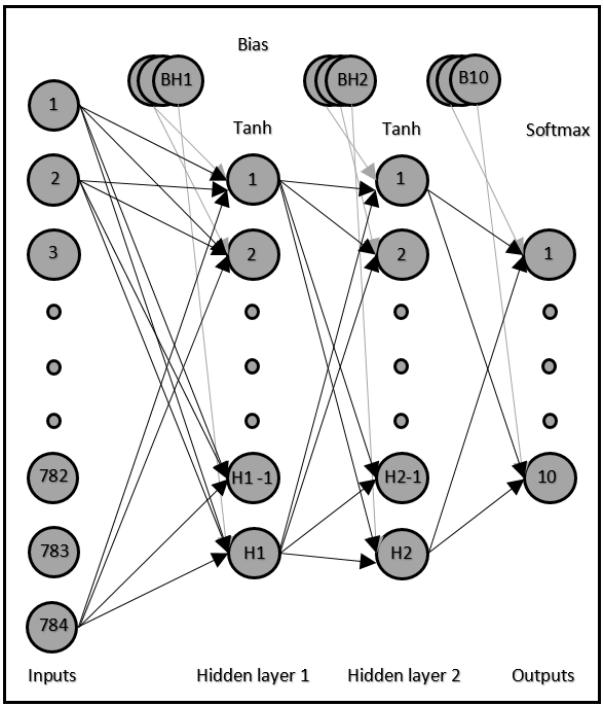


Figure 2.3: Architecture of the MNIST classifier

each bias and synapses. In our study those *prior* distributions were following a normal law with a mean value equal to zero. Their standard deviation were *hyperparameters* that influence the quality of the training. The aim of the algorithm implemented in PyMC3 are then to find an estimation of the *posterior* distribution $p(f|x, y)$ given training data x_i and their label y_i. Knowing this estimation, composed of several scalar for each bias and synapses representing a distribution, each new input data will give an output distribution. One can notice that by defining the number of scalars used to represent the distribution, the space of function has again a finite number of dimension. The output with the greater mean value is the prediction of the circuit. The algorithm used in our model is called NUTS it is an optimisation of the famous MCMC algorithm, it has been shown to be good choice considering the tradoff between efficiency and user friendliness [13]. At each step of the MCMC sampling, a new distribution is proposed and depending on its likelihood and its resemblance to our prior belief, the new distribution is either accepted or rejected [1]. The mathematics behind those algorithm are complicated, only the main ideas were investigated since the using of the algorithm does not requires a deep understanding. This bottom up presentation should have provides an global understanding of the theoretical object that are manipulated in this report and how they can be built. The next part

will focus on the experimental datas that were provided by the laboratory which are the building blocks of this work.

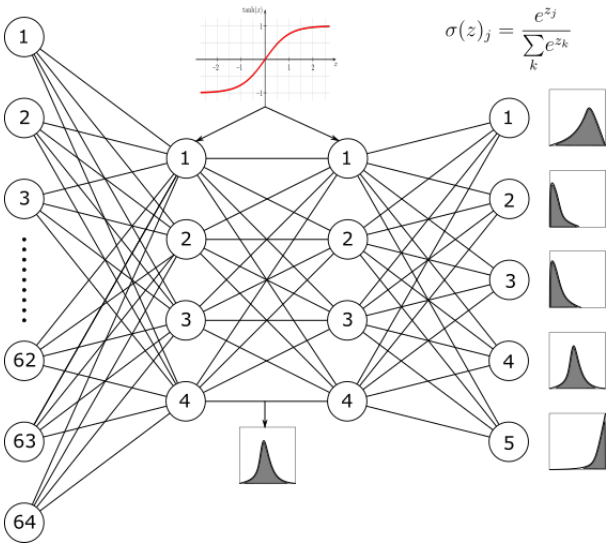


Figure 2.4: Architecture of the Bayesian neural network

in two different ways, one that gives a linear separation between the bins and one that is optimized in terms of programming energy consumption. The last one is done by optimising the separation between the bins and their authorised width depending of the controlability of the conductance. In other words this methods authorised more degrees of freedom to the distributions so that the devices properties can be better dealt with. Following this strategy the number of iteration needed to have separated bins is smaller. The less iterations means the shorter the time of programming and the lower the energy consumption. The algorithms were applied to the array to produced different number of level inside the programmable window. One specificity compared to [8] is that the RESET state (first narrow bin at the left of each plot) is also used. This state allows to get better zeros for the weights and gives one more states easily separable from the others. To each states correspond a compliance current. One of the main part of this study was to assert the efficiency of those different programming concerning the accuracy of neural network using RRAM to implemment synapses. As well as the effect of the conductance distribution time dependency. The following part explain the transition between the initial brut data and the accuracy result, which is the heart of the simulation strategy.

References

[1] T. Dalgaty, N. Castellani, D. Querlioz, *et al.*, "In-situ learning harnessing intrinsic resistive memory variability through markov chain monte carlo sampling," (2020).

[2] C. Yakopcic, R. Hasan, and T. M. Taha, "Memristor based neuromorphic circuit for ex-situ training of multi-layer neural network algorithms," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–7 (2015).

[3] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: materials, devices and applications," *Nanotechnology* **24**, 382001 (2013).

[4] R. Hasan, C. Yakopcic, and T. M. Taha, "Ex-situ training of dense memristor crossbar for neuromorphic applications," in *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH´15)*, 75–81 (2015).

[5] C. Li, D. Belkin, Y. Li, *et al.*, "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature Communications* **9**, 2385 (2018).

[6] P. Chen, B. Lin, I. Wang, *et al.*, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 194–199 (2015).

[7] G. W. Burr, R. M. Shelby, A. Sebastian, *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X* **2**(1), 89–124 (2017).

[8] P. Yao, H. Wu, B. Gao, *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature* **577**, 641–646 (2020).

[9] V. Milo, C. Zambelli, P. Olivo, *et al.*, "Multilevel hfo2-based rram devices for low-power neuromorphic networks," *APL Materials* **7**(8), 081120 (2019).

[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints* , arXiv:1412.6980 (2014).

[11] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *ArXiv* **abs/1506.02158** (2015).

[12] J. Zhao, X. Liu, S. He, *et al.*, "Probabilistic inference of bayesian neural networks with generalized expectation propagation," *Neurocomputing* **412**, 392 – 398 (2020).

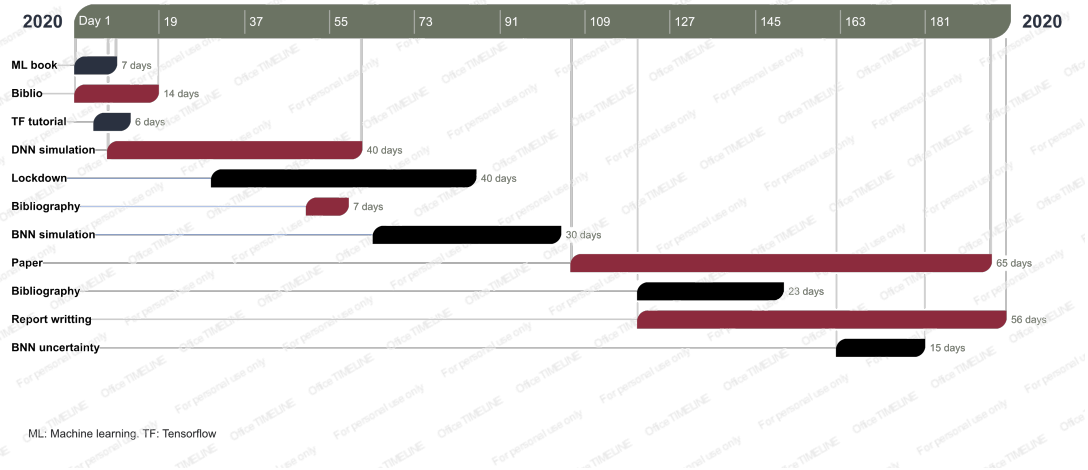
[13] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo," (2011).

[14] F. Choudry, E. Fiesler, A. Choudry, *et al.*, "A weight discretization paradigm for optical neural networks," in *in Proceedings of the International Congress on Optical Science and Engineering*, 164–173, SPIE (1990).

[15] S. Ambrogio, M. Gallot, K. Spoon, *et al.*, "Reducing the impact of phase-change memory conductance drift on the inference of large-scale hardware neural networks," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 6.1.1–6.1.4 (2019).

Appendixe

.1 Gantt diagram



.2 Cost evaluation

The remuneration provided by the CEA was 1511€ per month (counting all taxes) thus 9066€ in 6 month . Added to that 767€ as a bonus. Due to the lockdown the plan of the internship was modified and the only material I needed was the computer and the data provided by the CEA-LETI. However, all the simulations have been fully calibrated thanks to the RRAM arrays fabricated and tested at Leti in the framework of other projects. The cost for one RRAM lots is about 150keuros.