Apprentissage d'Opérateur Neuro-Inspirés Implémentés sur Nano-Composants pour le Traitement Spatial des Images

Olivier Brousse[1], Michel Paindavoine[1], Christian Gamrat[2]

[1]LEAD UMR CNRS 5022 Univ. Bourgogne
21065 Dijon, France

[2]CEA-LIST
Saclay, France

olivier.brousse@u-bourgogne.fr, michel.paindavoine@u-bourgogne.fr, christian.gamrat@cea.fr

Résumé − L'étude des nano-composants nous permet de les envisager comme des composants essentiels dans les systèmes de calculs du futur. Nous avons donc choisi d'étudier pour quels types d'applications ces composants seraient du plus grand intérêt. Dans cet article nous présentons les résultats d'une méthode synapto-inspirée permettant de configurer des composants memristifs pour implémenter des fonctions de traitement des images. Ce genre d'implémentation pourrait à terme prendre place au plus proche du capteur et ainsi accélérer les traitements d'images bas niveau.

Abstract − As nanoscale memristive devices are under studies, we choose to investigate in which application domain such components may be of the most interest. In this paper we present how the synapse-like behavior of such devices can be used to implement functions on nanoscale-based components. This method has been tested in the image processing application field in the theoretical case where processing are embedded at the sensor level.

1 Introduction

les systèmes de vision intelligents sont de plus en plus utilisés dans les systèmes embarqués. Néanmoins, leurs applications sont limitées dans le sens où les capacités de calcule des systèmes embarqués sont en deçà des besoins calculatoires pouvant dépasser 100 Gops, et ce dans ce contexte de basse consommation énergétique – typiquement 500mw – et de faible encombrement sur silicium – typiquement 1mm^2 pour les traitements implémentés au sein du capteur –.

Parallèlement à ce défi – d'implémenter des algorithmes de plus en plus complexes dans des architectures embarquées –, l'essor actuel des nanotechnologies,nous permet d'envisager des solutions alternatives au problème posé ci-dessus de par leurs promesses d'une plus grande capacité d'intégration des composants et leurs caractéristiques intrinsèques. Une solution prometteuse, proposée dans [1], est d'utiliser les caractéristiques proches des synapses biologiques des composants memristifs et de développer une approche neuro-inspirée permettant l'apprentissage de fonctions sur ces nano-composants.

Dans l'objectif de tester et valider ces nouvelles technologies et approches, nous avons choisi de les appliquer au domaine du traitement des images. La section 2 présente brièvement l'approche proposée. Les Sections 3 et 4 présentent deux cas de simulations nous permettant de valider de façon comportementale l'approche proposée au travers de deux exemples : une variante gros grain implémentant un filtre de détection de contours et une variante à grain plus fin reposant sur l'implémentation d'opérateurs arithmétiques tel que la multiplication-accumulation.Dans la Section 5 nous présentons finalement quelques conclusions et perspectives concernant cette approche et son intérêt dans le domaine du traitement des images.

2 Configuration Neuro-inspirée

En se basant sur des circuits d'apprentissages supervisés les auteurs de [2], [3] et [4] ont démontrés qu'il était possible de configurer des nano-composants memristifs structurés en "crossbar" à la manière de réseaux de neurones – un croisement entre 1 ligne et 1 colonne étant composé d'un composant memristif est assimilé à une synapse –. Il a ainsi été démontré, via des simulations électriques de modèles compacts et comportementaux, qu'il était est possible d'apprendre des fonctions Booléennes sur des composants memristifs.

Dans le domaine du traitement des images, les réseaux de neurones sont usuellement employés à des fin de détection, reconnaissance ou encore de classification. Néanmoins la grande flexibilité offerte par ces techniques permet de les utiliser pour nombre d'autres applications. Les

travaux présentés dans cet article reposent sur la possibilité de configurer, via un apprentissage, un ensemble de nano-composants pour lui faire effectuer une fonction cible.

Pour parvenir à ce résultat, nous devons savoir si la function cible est linéairement séparable ou non. Dans le premier cas, la configuration peut se faire directement sur le substrat matériel grâce à la règle du *delta binaire* [5] et ce de la manière présentée dans [6]. Dans le cas contraire, le réseau de neurones nécessaire à l'implémentation de la fonction doit comporter au minimum une couche cachée. De manière à pouvoir implémenter ce réseau à couches multiples à l'aide de l'apprentissage à *delta binaire*, nous devons déterminer les fonctions de la ou des couches cachées. Ceci est réalisé grâce à l'étude des fonctions apprises dans les couche cachées du réseau de neurones hors ligne. Ce qui nous permet d'identifier des fonctions linéairement séparables qui une fois apprises par le substrat peuvent être combinées pour donner la fonction cible.

La Figure 1 résume ceci en prenant l'exemple d'une fonction logique *Xor* à 8 entrées. L'étape hors ligne est réalisée en utilisant l'algorithme à rétro-propagation du gradient Rprop [7], le processus complet est décrit plus en détail dans [6].

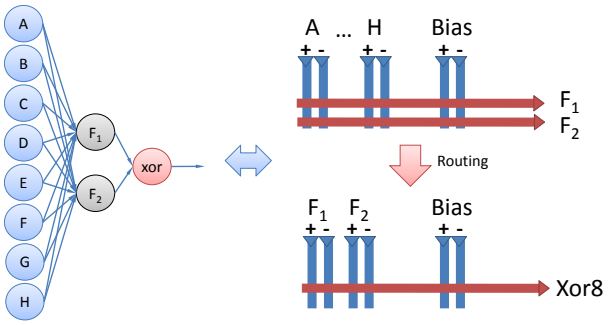


FIGURE 1 – Vue schématique d'un réseau permettant d'implémenter la fonction *Xor* avec 8 entrées en utilisant 2 unités cachées.

L'identification des fonctions cachées se faisant hors ligne, chacune d'entre elle est ensuite apprise sur le substrat. Une fois les fonctions cachées implémentées, les fonctions cibles peuvent à leur tour être apprises par le substrat. L'apprentissage matériel se fait en ajustant les conductances des nano-composants entre les couches successives et ce comme décrit dans [6].

3 Traitement d'images Binaire

Dans cette section, nous présentons un exemple simple de traitement d'image via notre architecture neuro-inspirée à nano-composant et ce en tant que preuve de concept. Cet exemple se base sur le filtre de détection de contours de Sobel opérant sur un voisinage de 3×3 pixels.

Le filtre de Sobel est un classique parmi les détecteurs de contours [?]. Cette approche utilise deux masques de convolution 3×3 pour détecter les contours verticaux et horizontaux dans les images. Les opérations équivalente en chaque pixel de l'image traitée sont rappelés dans les Équations 1 et 2.

$$\begin{aligned} Sobel_{Vert}(P_1,\ldots,P_9) &= |-P_1-2\times P_2-P_3 \\ &\quad +P_7+2\times P_8+P_9| \quad (1) \\ Sobel_{Hor}(P_1,\ldots,P_9) &= |-P_1-2\times P_4-P_7 \\ &\quad +P_3+2\times P_6+P_9| \quad (2) \end{aligned}$$

Où $P_1,\ldots,P_9$ correspondent aux valeurs des pixels. Le pixel courant P_5 étant inutile pour cette fonction 8 entrées seront utilisées par notre réseau. Pour cet exemple une adaptation binaire de ces filtres a été réalisée. Nous normalisons la somme des Équations 1 et 2 puis un seuillage est effectué pour n'obtenir que deux valeurs possibles. Dans le cas de l'implémentation sur notre cible, un réseau à une couche cachée contenant 3 unités a été utilisé pour nous permettre d'apprendre le filtre de Sobel comme représenté en Figure 2.

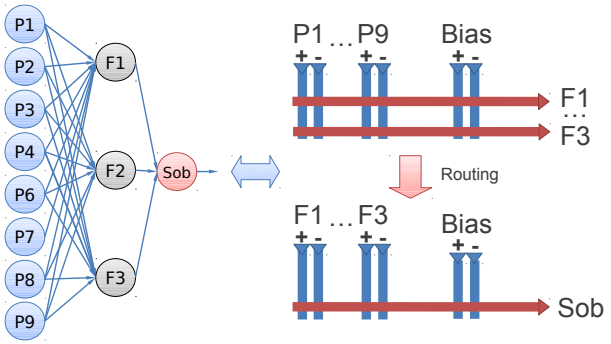
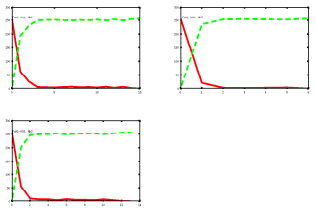


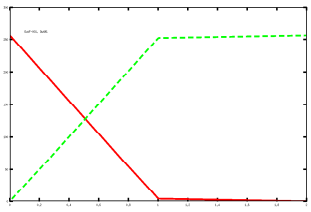
FIGURE 2 – Réseau de neurones permettant l'apprentissage du filtre de Sobel et son équivalent matériel

Comme introduit en Section 2, cette première étape d'apprentissage permet d'isoler via le unités cachées des fonctions linéairement séparables participant à la fonction de Sobel. Ces fonctions ont donc été utilisées dans la seconde étape permettant la modification des conductances des composants memristifs. La Figure 3 présente les résultats obtenus. La Figure 3(a) montre la convergence de l'apprentissage pour chacune des unitées cachées. La Figure 3(b) montre quant à elle la convergence l'apprentissage de la fonction de Sobel. Les Figures 3(c) et 3(d) présentent respectivement les histogrammes des valeurs prises par les conductances des couches cachée et de sortie pour cette simulation comportementale.

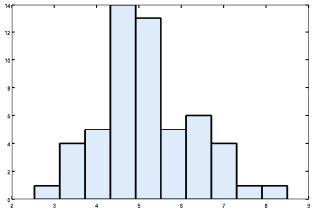
Pour cette simulation en vu d'une implémentation du filtre de Sobel, un total de 62 nano-composants sont requis. 54 de ces nano-composants permettent de réaliser la couche cachée – 8 entrées vers 3 unités cachées –, les 8 derniers étant dédiés à l'implémentation de la couche



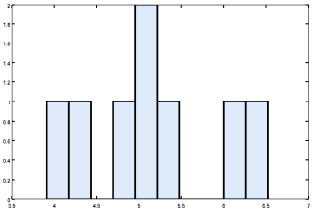
(a) Couche cachée : Convergence de l'apprentissage



(b) Fonction cible : Convergence de l'apprentissage



(c) Couche cachée : Histogramme des valeurs de conductances



(d) Couche de sortie : Histogramme des valeurs de conductances

FIGURE 3 – Simulation comportementale de l'implémentation du filtre de Sobel

de sortie. La Figure 4 présente à titre d'exemple 2 images traitées (en simulation) par notre filtre de Sobel. La partie supérieure correspond aux images d'origines et la partie inférieure aux résultats des traitements simulés.

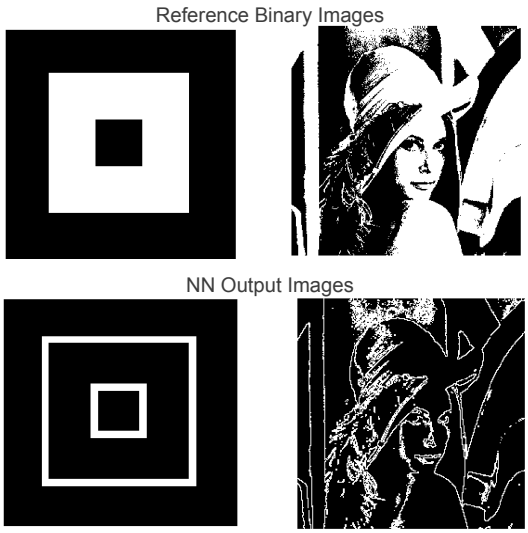


FIGURE 4 – Résultats de simulation du traitement d'image par notre filtre de Sobel binaire sur nano-composants

Dans le cas présenté ci-dessus, les images résultantes sont conformes à leur équivalent logiciel classique ramené au cas binaire. Nous pouvons donc conclure qu'il est possible d'implémenter, sur des substrats de nano-composants, des fonctions simple de traitement d'image grâce à l'approche proposée. Et ce même dans le cas où la fonction cible est non linéairement séparable.

4 Opérateur MAC : approche grain fin

Le principal problème avec l'approche présentée précédemment tient au fait que le nombre et/ou la précision des pixels (i.e. nombre de bits par pixels) doivent rester petits (i.e. < 16) pour que la technique soit utilisable. En effet, il est difficile de faire complétement apprendre à un réseau de neurones artificiels – dans le cas d'un apprentissage à 100% – des fonctions de tailles importantes par exemple avec $2^{25\times 8}$ possibilités – ce qui correspond au nombre de solutions définie en sorties d'une convolution avec un masque 5×5 avec des pixels de 8 bits –.

De manière à gérer cette limitation forte, nous proposons une approche à grain plus fin se basant sur l'implémentation de fonctions simple pouvant être mises en cascade. Ceci nous permet d'implémenter les fonctions usuellement utilisées en traitement des images. A titre d'exemple, cette section propose une implémentation de l'opérateur de multiplication-accumulation (MAC) qui est à la base de l'opération de convolution. Cette opération étant largement utilisée en traitement d'images, cet exemple montre la pertinence de l'approche proposée pour ce domaine d'application.

L'opérateur MAC est dans cette approche composé d'un multiplieur et d'un accumulateur comme présentée en Figure 5. Ces 2 éléments sont étant eux mêmes construits à partir de fonctions ET logiques et d'additionneurs complets comme présenté en Figure 6.

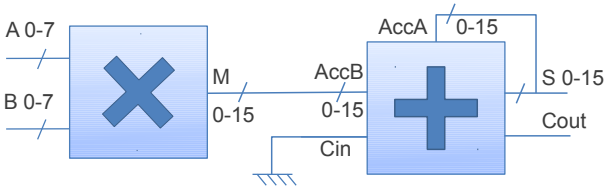
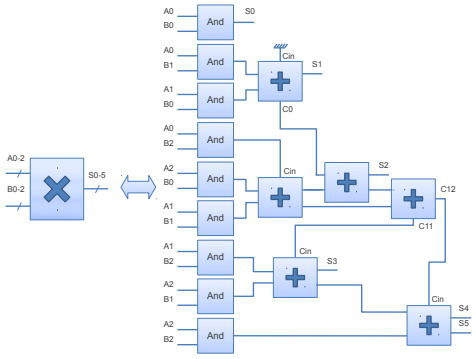


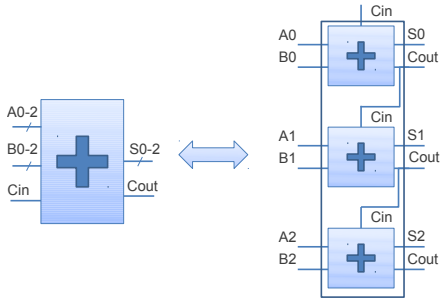
FIGURE 5 – Aperçu de haut niveau de la structure de l'opérateur MAC

L'opérateur MAC proposé ici peut être utilisé dans de nombreux cas de traitement d'images comme par exemple les convolutions à noyaux large – e.g 15×15 ou 25×25 – et ce avec différentes précisions au niveau des pixels. Ceci est possible car l'opérateur est constitué de cellules logiques simples et structurées pour obtenir un opérateur plus évolué.

L'exemple proposé en figure 5 nécessite de fait 3776 nano-composants pour implémenter les 64 ET logique – 6 conductances chacun –, les 68 additionneurs 1 bit – 40 conductances chacun – et les 28 additionneurs sans "retenue d'entrée" – 24 conductances chacun –. Ce dernier opérateur étant utilisé dans les cas ou la retenue d'entrée



(a) Structure d'un multiplieur à base de portes ET et d'additionneurs 1 bit



(b) Structure d'un additionneur à propagation de retenue

FIGURE 6 – Structures des opérateurs arithmétiques la précision est volontairement de 3 bits par soucis de lisibilité

n'est pas utile, ce qui nous permet de diminuer quelque peu le nombre de conductances nécessaires.

Ces 160 fonctions élémentaires sont apprises sur différents crossbar de nano-composants et structurées à l'aide d'un routage adapté. l'architecture complète est en cours de développement et a été brièvement introduite dans [6]. Ceci explique le peu de résultats quantitatifs présentés ici ce qui n'enlève rien à son intérêt en tant que preuve de concept.

5 Conclusion

Dans cet article nous avons montré qu'utiliser des nano-composants pour implémenter, grâce à un apprentissage matériel, des fonctions de base de traitement des images peut être intéressant tant au niveau du nombre réduit de composants nécessaire que de la flexibilité offerte par ce type de configuration.

La limitation remarquée dans le cas de fonctions plus complexes peut être contournée en utilisant une approche à grain plus fin qui utilise un assemblage de fonctions élémentaires pour fournir la fonction désirée. Il est ainsi possible d'implémenter des opérateurs arithmétiques dédiés au traitement des images à l'aide de nano-composants comme le montre les résultats prometteurs présentés ci dessus.

En perspectives de ces travaux, nous travaillons maintenant sur la conception d'une architecture, basée sur ces composants memristif, qui nous permettrait d'implémenter physiquement ces fonctions. Au niveau des simulations, nous nous penchons sur des cas de traitements d'images en niveaux de gris en utilisant l'approche grain fin de manière à implémenter différents filtre de tailles variées allant de 5×5 – e.g. filtre Gaussien – à 25×25 – e.g. de-convolutions d'images –. Ces derniers filtres pourraient être implémentés au plus près du capteur et ainsi augmenter la qualité des images en éliminant le bruit d'acquisition du capteur.

Références

[1] M. He, J.-O. Klein, and E. Belhaire, "Design and electrical simulation of on-chip neural learning based on nanocomponents," *Electronics Letters*, vol. 44, April 2008.

[2] J.-M. Retrouvey, O. Klein, J, Y. Liao, S, and C. Maneux, "Electrical simulation of learning stage in ogcntfet based neural crossbar," in *DTIS 2010 Conference*, March 2010, hammamet, Tunisia.

[3] D. Chabi and J.-O. Klein, "Hight fault tolerance in neural crossbar," in *Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*. IEEE, Mar 2010, pp. 1–6. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5487552

[4] O. Brousse, "A bio-inspired agent-based programming environment for pervasive platforms," Ph.D. dissertation, Univ. Montpellier 2 / UNIL, 2010.

[5] I. Russell, "Neural networks theory and applications," *The Journal of Undergraduate Mathematics and its Applications (UMAP)*, 1993.

[6] D. Chabi, O. Brousse, J.-O. Klein, and M. Paindavoine, "Learning arithmetic operators in nano-devices," *Submited to IEEE Transaction on Neural Networks.*, 2011.

[7] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning : The rprop algorithm," in *Proceedings of the IEEE International Conference on Neural Networks*. IEEE Press, 1993, pp. 586–591.

Acknowledgment

Les auteurs aimeraient remercier l'ANR qui à financé le projet PANINI (ANR-07-ARFU-008) ainsi que nos partenaires pour toutes les discussions fructueuses qui ont contribuées à l'obtention de ces résultats.