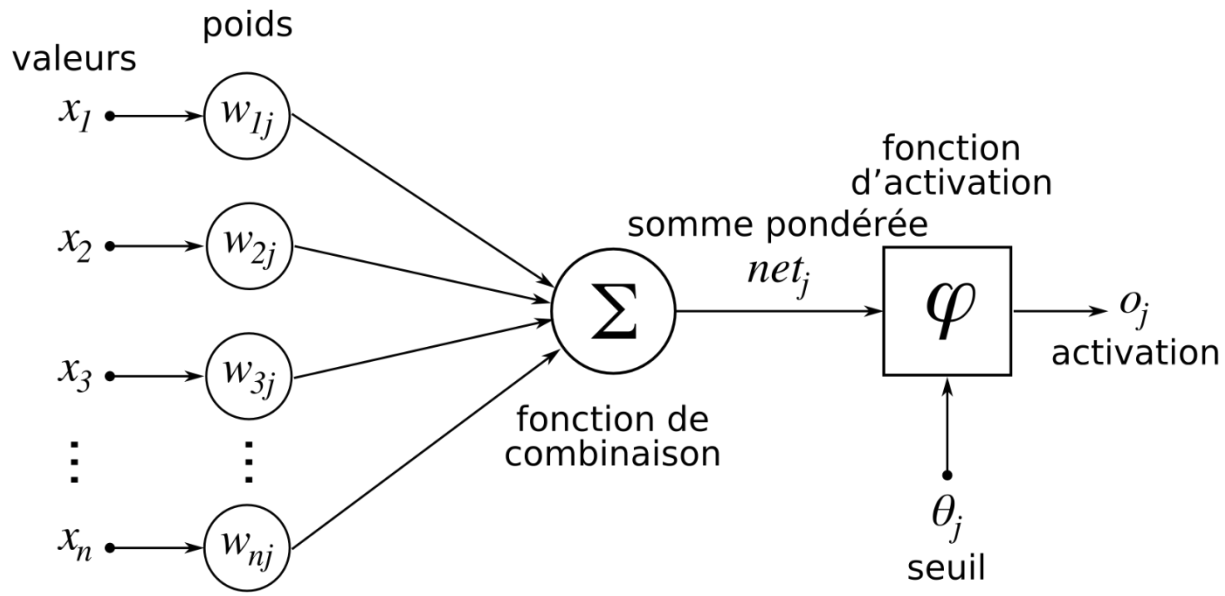
LUT Brain 2020 (v1.0)

1. Introduction

Les réseaux de neurones artificiels (RNA) forment un modèle de calcul s'inspirant du fonctionnement du cerveau. Ces réseaux peuvent être entrainés pour réaliser différentes tâches typiquement difficiles à réaliser au moyen des méthodes de programmations traditionnelles, telles que la reconnaissance de forme, la reconnaissance vocale, l'apprentissage, et d'autres. Néanmoins, la réalisation de RNA implique généralement un nombre impressionnant de calculs sur des nombres réels. Toutefois, des recherches récentes montrent qu'il serait possible d'obtenir de très bons résultats avec seulement des nombres ternaires (-1, 0, 1), voir même des nombres binaires (0, 1). Vous trouverez une petite introduction aux RNA ci-dessous :

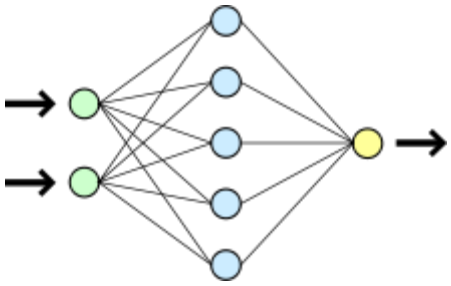
https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels

On y trouve notamment la description suivante d'un neurone artificiel :



https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels#/media/File:ArtificialNeuronModel_francais.png

Chaque entrée est multipliée par un poids (W_{ij}). Ensuite, on calcule la somme de toutes ces entrées pondérées par leurs poids respectifs. Finalement on applique une fonction non linéaire (φ), qui peut dépendre d'un seuil (θ_j). Le résultat (o_j) s'appelle l'activation. On peut s'en servir comme sortie du réseau, ou comme signal intermédiaire qui sera utilisé à l'entrée des neurones de la couche suivante :



https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels#/media/File:Neural_network.svg

De récents travaux ont montré que les poids et les activations pouvaient être des nombres binaires (+1 ou -1). Dans ce contexte, chaque neurone devient en fait une fonction booléenne de ses entrées. Dans le cadre de ce projet, on va remplacer chaque neurone par une Look-up table (LUT) avec un nombre restreint d'entrées (maximum 8). Il vous est demandé de mettre au point une plate-forme embarquée optimisée permettant l'exécution rapide de RNA pré-entrainés en logique binaire basés sur des LUT. Les RNA considérés sont des réseaux de perceptrons à plusieurs couches qui prendront en entrée des images en noir et blanc. Le système à réaliser doit être en mesure de calculer rapidement la (les) sortie(s) d'un RNA constitué de N couches de perceptrons.

Vous ne partez pas les mains vides, une plate-forme matérielle de base ciblant la carte De-10, ainsi qu'une application logicielle de base vous sont fournies. Des RNA de différentes tailles comportant différents nombres de couches ainsi que des images d'entrées peuvent être générés aléatoirement pour faciliter le développement.

2. Application logicielle de base

Les fonctions et les classes permettant d'appliquer une machine neuronale à des images ont déjà été complétées. Des configurations particulières pour le test vous seront données au moment opportun.

Fichiers	Description
Main.cpp	Programme principal. Exécute un réseau de neurones de deux couches généré aléatoirement.
NN .h/.cpp	Classe représentant une réseau de neurones à plusieurs couches.
NNLayer .h/.cpp	Classe représentant une couche (de neurones) d'un réseau de neurones.
Image .h/.cpp	Classe permettant de représenter les images qui seront utilisées en entrée des réseaux de neurones.
VGA .h/.cpp	Classe qui gère l'affichage

3. Spécification de la plate-forme matérielle de base

La plate-forme matérielle de base comprend un processeur Nios II/f doté d'un contrôleur mémoire SDRAM, un contrôleur VGA, ainsi que quelques autres périphériques usuels.

4. Consignes

Votre but est d'accélérer au maximum l'évaluation d'un RNA de taille quelconque basé sur des LUT. Ce réseau sera appliqué à toutes les sous-images d'une image de départ pour produire une image résultat (comme pour une convolution). Par exemple, si on a une image de 100x100 et des sous-images de 10x10, on devra évaluer 91x91 = 8281 sous-images avec le RNA, chacune créant un pixel de l'image résultat qui aura une taille de 81x81.

Vous devez optimiser la vitesse de calcul dans deux **scénarios** différents :

a) Vous connaissez les LUT 24h à l'avance (il vous est donc possible de recompiler certaines parties de l'application à l'avance ainsi que certains IP dans le FPGA)
b) Vous connaissez les poids ternaires et les seuils 5 minutes à l'avance (il vous est donc possible de recompiler certaines parties de l'application à l'avance mais pas les IP dans le FPGA)

Vous DEVEZ utiliser les techniques suivantes :

- Au moins une instruction spécialisée multi-cycles réalisée par vous-même.
- Au moins un cœur spécialisé sur le bus Avalon réalisé par vous-même.

Vous pouvez utiliser les techniques suivantes :

- Réutilisation d'IP logiciel et matériel (en citant la source)
- Réalisation d'un système multiprocesseur
- Utilisation du processeur embarqué ARM (obligatoire si vous aspirez à la note A)

Veillez toutefois à ce que votre contribution personnelle soit suffisamment significative pour évaluer votre maitrise des techniques. Pour chaque ressource matérielle que vous implémentez vous-même, vous fournirez également l'ASM correspondant **et son chemin de données**.

4.1. Formation des équipes

Sauf exception, les équipes de projet seront formées de 3 étudiants. La formation des équipes est laissée libre. Toutefois, toutes les équipes devront obligatoirement être constituées au moment de la présentation du projet. **Attention, chaque équipier sera notamment responsable d'un circuit à écrire en ASM/VHDL, ou d'une architecture multiprocesseur ou encore de l'utilisation du processeur ARM embarqué.**

4.2. Répartition des tâches

Les membres de l'équipe devront se répartir les différentes tâches associées au projet. Un des membres sera identifié comme responsable d'équipe et aura comme responsabilité particulière de gérer le bon déroulement du projet jusqu'à son achèvement. Lors de la première revue de projet, qui aura lieu une semaine après la présentation du projet, le responsable d'équipe devra remettre une liste de répartition de tâches signée par tous les membres de l'équipe.

4.3 Rappel du calendrier

16	17 B2	18 **C6** B2	19 **P1** B1	20 B1	21 B1	22
23	24 B1	25 **Intra** B1	26 **P2** B2	27 B2	28 B2	29
1 **MARS**	2	3	4	5	6	7
	PÉRIODE DE RELÂCHE					
8	9 B2	10 **P3** B2	11 **P4** B1	12 B1	13 B1	14
15	16 **P5** B1	17 B1	18 **P6** B2	19 B2	20 B2	21
22	23 B2	24 **P7 (étape)** B2	25 **P8** B1	26 B1	27 B1	28
29 **AVRIL**	30 **P9** B1	31 B1	1 **P10** B2	2 B2	3 B2	4
5	6 **P11** B2	7 B2	8 **P12** B1	9 B1	10	11
12	13	14 **P13 (démo)** B1	15 Journée de lundi B1	16 Journée de vendredi B1	17 **Projets intégrateurs**	18

5. Revue de projet

Chaque semaine, le professeur rencontrera les équipes lors de la revue de projet. Ces rencontres auront lieu selon l'horaire du cours. L'objectif de cette revue est de faire le point sur l'avancement du projet. Les responsables d'équipe devront faire état de l'avancement du projet, du suivi de la planification des tâches,

des difficultés rencontrées dans la réalisation du projet et des solutions envisagées. Tous les membres de l'équipe doivent être présents. Les membres évalueront également le travail de leurs pairs au moyen d'une lettre parmi les suivantes : A (supérieur aux attentes), B (conforme aux attentes), C (inférieur aux attentes) D (très inférieur aux attentes, voire nul). Le professeur se réserve le droit de modifier la note finale des étudiants en fonction du travail individuel de chacun. Les objectifs à atteindre pour les 3 premières semaines sont les suivants :

Semaine 1 (P2) : Avoir compris le code dans ses moindres détails, supprimé l'utilisation de la virgule flottante, profilé le code, et identifié les sections critiques à améliorer.

Semaine 2 (P3) : Avoir optimisé le code C au maximum et réécrit chaque fonction destinée à être implantée en matériel sous la forme d'une fonction C indépendante.

Semaine 3 (P5) : Avoir écrit les ASM correspondants aux modules matériels et adapté les fonctions C pour qu'elles soient fidèles aux ASMs.

Semaine 3 (P7) : Avoir finalisé le rapport d'étape.

6. Le rapport d'étape

Le rapport d'étape devra faire état de l'avancement du projet. L'application doit être décomposée en un graphe de tâches et sous-tâches profilées de sorte que l'incidence de chaque tâche et sous tâche sur le temps total soit très explicite. Forts de cette analyse, vous proposerez une architecture générale optimisée pour laquelle les tâches ont été accélérées. Outre la description de cette architecture, on s'attend à ce que pour chaque tâche, vous donniez :

a) Si elle est réalisée en logiciel ou en matériel (ou mixte)
b) Des résultats de simulation des parties logicielles
c) Le code source des ASM (machines à états algorithmiques) des parties matérielles
d) Les procédures de test/vérification qui seront utilisées pour la validation
e) Une preuve que si toutes les accélérations locales sont atteintes, l'application au complet atteindra une vitesse de fonctionnement donnée.

Pondération :

1) Analyse des temps : **2 points**
   a) Les temps les plus significatifs sont bien repris
   b) Leurs influences sur les diverses fonctions sont bien montrées sous forme de graphe de dépendance
2) Accélérations proposées : **4 points**
   a) Les accélérations sont pertinentes et bien décrites pour chaque tâche
   b) Les accélérations matérielles sont réalistes et démontrées
   c) Les accélérations logicielles ont été testées
   d) Preuve est faite que si toutes les accélérations individuelles sont atteintes, la vitesse finale annoncée est atteinte.
3) Architecture et ASMs : **2 points**

a) L'architecture générale est réaliste et bien décrite.
b) Tous les ASMs sont décrits et corrects.

4) Plan de test : **1 point**
   1) Les plans de test sont réalistes et suffisants
5) Qualité de la présentation : **1 point**

7. La présentation Finale (20 points)

Sous forme de diaporama « Powerpoint », elle dure 15 minutes. Chacun doit participer. La présentation sera de type « technico-commerciale », c'est-à-dire que vous vous adressez à un public de spécialistes (typiquement des ingénieurs) mais votre but est de vendre votre produit. Vous devez donc :

- Vanter les mérites de votre approche :
  - Performance (**3 points**)
  - Vérification : montrez que votre système a été suffisamment testé (**3 points**)
- Mentionnez également les faiblesses connues et donnez des pistes d'améliorations (**2 points**)
- La qualité graphique de la présentation sera aussi évaluée (**2 points**)

10 points sont attribués à la qualité (et les performances) de la démonstration.

À la fin des présentations, chaque étudiant devra « acheter » un produit parmi ceux proposés par les équipes concurrentes et justifier pourquoi.

8. Le rapport écrit (20 points)

Votre but est de convaincre que vous maitrisez parfaitement tous les concepts vus au cours en les ayant appliqués dans votre projet. Outre l'introduction et la conclusion, le rapport doit comprendre deux grandes parties :

1) Une description générale de votre architecture et une mise en valeur de ses parties les plus ingénieuses et/ou performantes (**4 points**). Vous décrirez aussi :
   a. les modifications que vous avez du apporter à votre première spécification et pourquoi.
   b. Les implémentations originales et celles que vous avez réutilisées (**en citant les sources**). Attention, ne pas citer une source est considéré comme étant du plagiat.
2) Une documentation technique qui doit permettre à quelqu'un de s'approprier votre projet et d'y faire des modifications aisément (**4 points**). On y trouvera notamment :
   a. Les algorithmes utilisés.
   b. Les ASM et leur fonctionnement (avec un niveau de détail suffisant, variable selon la complexité). Vous donnerez également leur chemin de données.
   c. Les instructions spécialisées.

Dans la conclusion (**2 points**), vous mentionnerez notamment :

- Les apprentissages que vous avez réalisés et jugerez de leur pertinence.
- Les problèmes rencontrés qui vous ont fait perdre du temps inutilement.
- Toutes suggestions pour améliorer le cours et le projet.
- Si vous autorisez le prof à réutiliser votre travail.

10 points sont attribués à la qualité de l'architecture développée.

Les rapports finaux sont à déposer dans le casier prévu pour la remise des devoirs (local M-5405). Il ne faut pas oublier de mettre le sigle et le titre du cours. Il est impératif de respecter la date limite de remise du rapport (-1 pt / jour de retard).

9. Structure du CD/Clé USB à remettre en guise de rapport:

9.1 Préparation des sources

Créez un répertoire "ELE8307A2018" dans la racine de votre répertoire au laboratoire. Déposez-y toutes vos sources (logicielles et matérielles) en vous assurant que le contexte Eclipse est bien dans ce répertoire et compilez le projet. Assurez-vous qu'il est pleinement fonctionnel. Ce répertoire doit être présent le jour de la remise du rapport et ne plus être modifié ensuite.

Veillez à ce que ce point soit scrupuleusement respecté. Si votre code ne fonctionne pas, certaines parties de votre projet ne pourront pas être évaluées !

Nettoyez ensuite votre compte de tous les fichiers qui encombrent les serveurs inutilement.

9.2 Création de l'archive

Copiez le répertoire ELE8307A2018 à la racine du CD ou de la clé USB.

Créez ensuite un répertoire "doc" dans lequel vous déposerez :

1. votre présentation d'étape sous le nom "etape" (format préféré : PowerPoint)
2. votre présentation finale sous le nom "final" (format préféré : PowerPoint)
3. votre rapport sous le nom "rapport" (format préféré : pdf)
4. un fichier par lequel vous me donnez (on non) le droit de réutiliser votre code sous le nom de "readme.txt".

!!! RESPECTEZ BIEN TOUTES LES CONSIGNES SVP !!!