LES RÉSEAUX DE NEURONES MATÉRIELS

Mewtow,

09 avril 2015

Zeste de savoir

Oeste de savoir

Table des matières

1 Introduction

Les ordinateurs sont vraiment loin d'être les seuls systèmes capables de traiter de l'information. Les automates mécaniques furent les premiers à en être capables : les ancêtres des calculettes étaient bel et bien des automates basés sur des pièces mécaniques, et n'étaient pas programmables. Par la suite, ces automates furent remplacés par les circuits électroniques analogiques et numériques non-programmables. La suite logique fût l'introduction de la programmation : l'informatique était née.

De nos jours, de nouveaux types de circuits de traitement de l'information ont vu le jour. On peut citer des circuits électroniques programmables, dans lesquels on ne trouve aucune notion d'instruction : les CPLD et FPGA. On trouve aussi des systèmes non-programmables à base d'ADN, mais leur manque de programmation ainsi que leur spécialisation à des problèmes bien précis ne pas sont pas très encourageants pour du calcul généraliste. Mais d'autres circuits commencent à voir le jour : les **architectures neuromorphiques**, qui se basent sur des réseaux de neurones matériels.

Dans ce qui va suivre, nous allons voir :

- ce qu'est un réseau de neurones ;
- à quoi servent ces architectures, leurs avantages et leurs inconvénients ;
- ce qu'elles simulent : on va devoir faire des rappels sur les neurones et les synapses ;
- comment neurones et synapses sont implantées en matériel (on fera très vite) ;
- et comment sont organisées ces architectures : on va voir leur structure ;
- quels sont les projets en cours et l'état de la recherche sur le sujet.

[CPLD] : Complex Programmable Logic Device [FPGA] : Field Programmable Gate Array

2 Réseaux de neurones matériels

Ces architectures s'inspirent fortement du cerveau humain, et du fonctionnement du système nerveux. Ces réseaux de neurones sont capables d'apprendre presque seuls, à partir d'exemples. Dans certains cas, il faut donner des indications de catégorisation pour que le réseau de neurones arrive à faire son travail. On doit corriger certaines de ses erreurs de départ, pour lui donner une idée de ce qu'on recherche. On parle alors d'apprentissage **supervisé**.

Mais dans d'autres cas, le réseau de neurone peut apprendre tout seul à partir d'exemples, et est capable de catégoriser de lui-même les entrées qu'on lui fournit. Ce genre d'apprentissage automatique (on dit aussi **non-supervisé**) est très efficace pour une grande classe d'applications :

- catégorisation d'inputs ;
- reconnaissance des visages ;
- reconnaissance d'émotion ;
- les rétines artificielles ;
- les systèmes de traitement visuel ;
- vidéo-surveillance intelligente ;
- systèmes de contrôle de moteurs ;
- etc.

2.1 Simulation matérielle de systèmes nerveux

A départ, les neurones simulés étaient simples, et cela suffisait. Mais par la suite, ces simulations ont vu apparaître des neurones de plus en plus sophistiqués, et des réseaux de neurones plus complexes.

Mine de rien, la simulation logicielle d'un système nerveux, même simple, demande beaucoup de puissance de calcul. Par exemple, Markam, un des utilisateurs du supercalculateur Blue-gene a déclaré qu'il était capable de simuler seulement 50.000 neurones en temps réel biologique (à la même vitesse que ce qui se passe dans un vrai réseau de neurones). Pour simuler 1 millions de neurones, cela prendrait 8 à 10 fois plus de temps que le temps réel biologique, et serait réellement impraticable.

Pour accélérer ces calculs, certains ont eu l'idée de créer des réseaux de neurones matériels. Avec les technologies VLSI récentes, il est possible de créer des circuits électroniques capables de simuler relativement facilement des réseaux de neurones matériels. On peut ainsi créer des circuits électroniques qui fonctionnent comme un neurone : la tension et le courant à leurs bornes suit les mêmes équations que la tension de membrane d'un neurone. Même chose pour les synapses.

Simuler un système nerveux en matériel peut permettre d'étudier des systèmes nerveux d'animaux simples, la moelle épinière d'une souris, le cervelet d'un humain, etc, que la simulation logicielle ne peut pas traiter.

Ce n'est pas la première fois que des chip matériels sont crées pour accélérer des simulations très spécifiques. Il existe par exemple des systèmes matériels spécialement conçus pour les simulations physiques de champ électriques ou gravitationnels : le projet GRAPE en est un exemple. C'est ainsi : quand on veut de la performance, le logiciel est de toute façon trop lent, et l'informatique est dépassée.

Divers projets de ce type ont alors vu le jour. On peut notamment citer les projets BrainScaleS, qui est à l'origine d'un circuit VLSI de 200000 neurones et 49 millions de synapses.

On trouve aussi le projet Neurogrid du groupe de recherche "brain in silicon", de l'université de Stanford. Celui travaille sur la simulation d'une région du cortex, qui contient des neurones spécialisés : les neurones pyramidaux. Ce groupe de recherche a déjà crée une puce non-programmable : le câblage des synapses ne peut pas être modifié, mais les paramètres des synapses sont configurables.

Enfin, certains chercheurs du MIT ont créé un simulateur matériel simulant quelques neurones, dans un but de recherche fondamentale sur le comportement des récepteurs endo-cannabinoide de la membrane des neurones.

Plus connus, les projets Humain brain ou FACETS, financés par l'UE, sont des projets de simulation beaucoup plus récents, dont la presse a parlé.

2.2 Les accélérateurs matériels

Certains chercheurs se sont dit que cela pouvait aussi servir à créer des circuits programmables généralistes.

De nos jours, un problèmes se fait jour : les architectures informatiques, basées sur des processeurs programmables qui exécutent des instructions, posent de plus en plus problème. Qu'il s'agisse d'architectures Von-Neumann avec un Program Counter, d'architectures Dataflow, ou de machines plus complexes capables d'exécuter des langages fonctionnels ou logiques, deux problèmes restent récurrents :

- les instructions doivent être chargées depuis la mémoire, et décodées ;
- la mémoire est lente d'accès.

Et cela coûte cher non seulement en circuits - décoder des instructions et communiquer avec la mémoire demande des circuits adaptés-, mais aussi en temps d'exécution : le temps mis pour traiter les instructions, et rendre le système programmable, n'est pas négligeable.

Les circuits neuromorphiques ont l'avantage de ne pas exécuter d'instructions : tous leurs circuits sont dévolus au traitement de l'information. Et c'est sans compter l'absence de mémoire, et donc d'accès mémoires. De plus, ces circuits neuromorphiques peuvent parfois apprendre tout seuls, et ne nécessite donc pas de programmation explicite : ces circuits sont parfois capables de généraliser tous seuls à partir d'exemples.

En conséquence, certains chips ont étés inventés pour servir non de simulateur, mais de calculateur spécifique, comme accélérateurs matériels.

A ma connaissance, une entreprise du nom de CogniMem a déjà commercialisé un accélérateur neuromorphique comprenant trois couches de respectivement 256, 1024, et 16392 neurones. L'ensemble fonctionne à la fréquence de 27 Mhz.

Plus récemment, le projet Synapse d'IBM a fait grand bruit. Celui-ci a permis de créer une carte accélératrice qui se connecte sur PC via un port USB. Ce circuit contient 256 neurones, reliés entre eux par 262144 synapses : certaines de ces synapses sont programmables, d'autres non.

3 Neurones et synapses

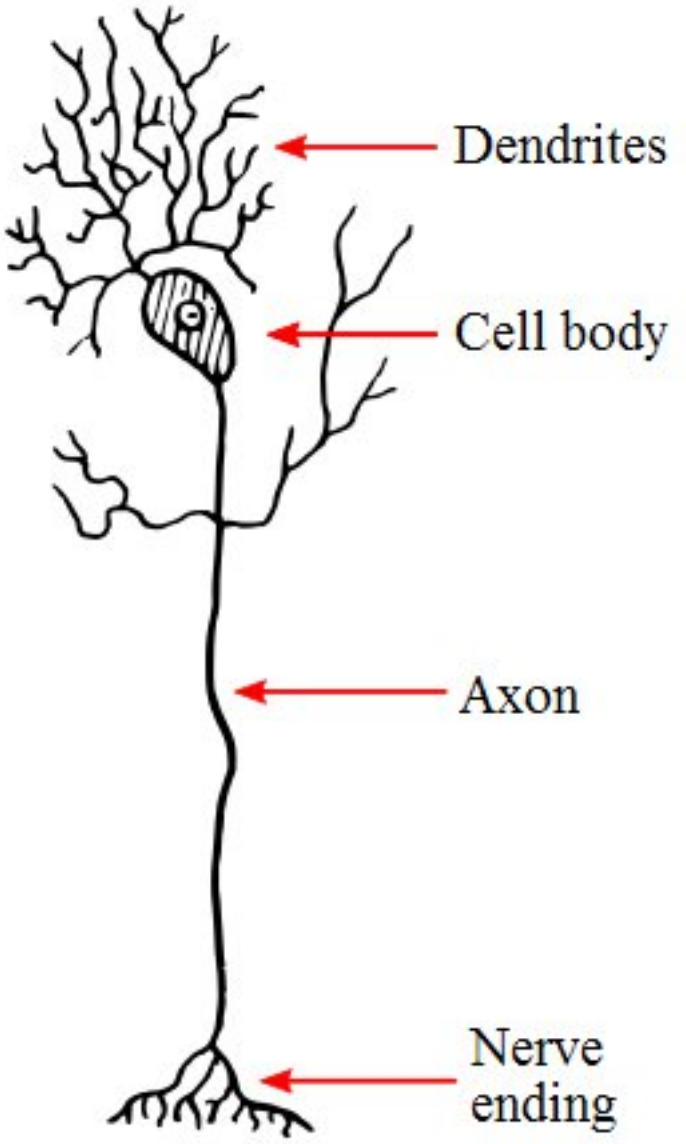
Comme je l'ai dit, ces architectures neuromorphiques simulent des neurones. Pour comprendre leur fonctionnement, nous sommes donc obligés de voir ce qu'est un neurone, une synapse, et comment ceux-ci fonctionnent. Préparez-vous : le cours de biologie arrive.

3.1 Structure d'un neurone

Les cellules du cerveau (et du système nerveux en général) sont toutefois un peu spéciales, et ont des particularités. Ces cellules sont ce qu'on appelle des **neurones**.

Il existe un grand nombre de neurones différents, qui se différencient par leur taille, leur forme, ou quelques autres paramètres. Mais tous ces neurones gardent malgré tout des points communs quant à leur forme.

Par exemple, un neurone ressemble presque toujours à ceci :



-> <-

En gros, il est composé de 4 parties :

- une dendrite sur laquelle arrivent les influx nerveux provenant d'autres neurones ;
- un corps cellulaire, qui se charge de traiter les influx nerveux et des tas d'autres trucs bien chimiques ;
- un axone, sorte de câble sur lequel l'influx nerveux crée par le neurone est propagé ;
- et un ensemble de boutons synaptiques, qui vont connecter l'axone sur une ou plusieurs dendrites.

Le système nerveux est constitué d'un grand nombre de neurones du genre, reliés entre eux par ce qu'on appelle des **synapses**. Ces synapses vont connecter l'axone d'un neurone aux dendrites d'un autre.

3.2 Fonctionnement d'un neurone

Ces neurones échangent des informations, codées par des influx nerveux, des sortes de "courants électriques", qui passent de neurones en neurones, et se propagent à travers tout le système ner-

veux. Ces influx nerveux sont ensuite traités par les neurones.

Ces influx nerveux ne sont pas toujours de courants électriques, mais correspondent le plus souvent à des variations de la tension électrique de la membrane du neurone. Lorsque le neurone ne fait rien, une tension d'environ -70 millivolts est présente sur sa membrane. Lorsqu'un influx nerveux passe, cette tension augmente localement.

Le traitement effectué par un neurone sur un influx nerveux est particulièrement basique. Le neurone fait la somme de tous les influx verveux qui arrivent sur ses dendrites à un moment donné. Si cette somme dépasse un seuil fixé à l'avance, alors celui-ci émet un nouvel influx nerveux sur son axone. Dans le cas contraire, il n'émet rien.

3.3 Plasticité synaptique

Les synapses ne sont pas des fixes : dans certaines situations, celles-ci peuvent se reconfigurer : de nouvelles synapses peuvent se créer, d'autres mourir, etc. De plus, l'efficacité de celle-ci à transmettre l'influx nerveux d'un neurone à un autre peut varier. L'ensemble de ces phénomènes s'appelle la **plasticité synaptique**.

Cette plasticité serait le mécanisme qui nous permettrait d'apprendre. On ne va pas rentrer dans le détail, mais il va de soit que simuler cette plasticité est primordiale pour obtenir des circuits neuromorphiques qui apprennent tous seuls : on va devoir implanter un modèle de plasticité, et des synapses dans nos circuits.

Pour faire simple, la majorité des phénomènes de plasticité synaptique sont des variantes de ce que l'on appelle la **règle de Hebb** : si deux neurones émettent des influx nerveux à peu-prêt en même temps, la synapse qui les relie se renforce.

D'autres formes de plasticité existent, et notamment la plasticité fonction du temps d'occurrence des impulsions, aussi appelée **Spike Timing Dependant Plasticity**, qui prend en compte le fait qu'un des deux neurones émet avant l'autre. Cette plasticité fait la différence entre le neurone émetteur de l'influx nerveux, et le récepteur.

Si l'émetteur émet avant le récepteur, la synapse se renforce. Si c'est le récepteur qui émet avant l'émetteur, la synapse s'affaiblit. Plus le temps entre les influx nerveux est long, moins l'effet renforçateur ou affaiblisseur sur la synapse est faible. Cette dernière forme est la plus utilisée dans les réseaux de neurone matériels.

Avec des formes simples de plasticité à long terme, on pourrait aussi rajouter la potentialisation à long terme, la dépression à long terme, la facilitation, l'habituation, la sensibilisation, et quelques autres. Mais celles-ci ne sont pas encore implantées en matériel.

4 Implantation matérielle

Il est maintenant temps de voir comment ces neurones sont traduits en matériel.

Simuler le véritable comportement d'un neurone ou d'une synapse couterait vraiment beaucoup de circuits. A la place, les concepteurs de simulateurs logiciels et matériels utilisent des neurones simplifiés, dont la relation entre tension et courant (influx nerveux) est décrite par une équation relativement simple, et surtout approximative.

Une liste détaillée de ces modèles peut être trouvée ici :

- Modèles de neurones biologiques ;
- Spiking neuron models

Le premier type de neurone se base sur un principe relativement simple : il modélise le principe qui est derrière la conduction de l'influx nerveux.

Pour commencer, il faut savoir que l'influx nerveux n'est pas, comme on pourrait le croire, un courant électrique qui parcours l'axone et les dendrites. Les axones et les dendrites ne sont pas des câbles électriques. A la place, cet influx nerveux consiste en une onde de tension, qui se propage sur des segments d'axone, et éventuellement de dendrites (dans la suite, je ne parlerais que d'axone par simplicité).

Mais d'où sort cette onde de tension ? Simple : le neurone est une cellule comme une autre, dans laquelle l'intérieur et l'extérieur sont séparés par une membrane : la membrane plasmique. Des deux cotés de cette membrane, on trouve respectivement le milieu intra-cellulaire (dans la cellule), et extra-cellulaire (en-dehors de la cellule). Ces deux milieux sont composés d'une solution qui mélange de l'eau, diverses molécules, et surtout : quelques ions, des atomes de charge non-neutre. En conséquence, chacun de ces milieux est relativement conducteur.

L'ensemble forme donc un ensemble formé de deux milieux conducteurs, séparés par une membrane isolante. Deux morceaux de conducteur séparés par un isolant, cela ressemble fortement à un composant électronique que l'on nomme le condensateur. Conséquence : la membrane se comporte comme un condensateur.

Des deux cotés de la membrane, les quantités d'ions ne sont pas les mêmes. Par exemple, les quantités de calcium, potassium, et sodium, changent suivant le côté de la membrane où l'on se place. Cela vient du fait que la membrane est percée par de nombreuses molécules qui servent de **pompes**.

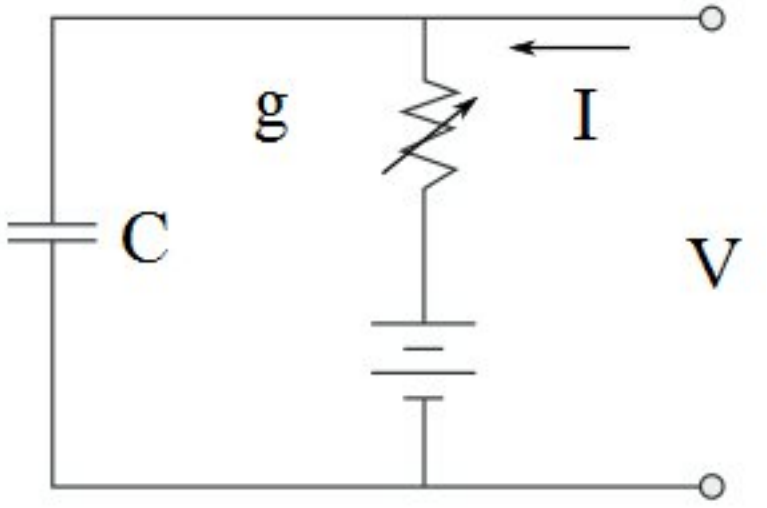
Ces pompes vont expulser les ions d'un coté de la membrane vers l'autre coté. Ces pompes vont fonctionner jusqu'à ce que la concentration en ions de l'autre coté de la membrane devienne trop forte. En conséquence, l'intérieur et l'extérieur sont chargés différemment, donnant naissance à une tension de repos, d'environ -70 millivolts. Ces pompes peuvent donc se modéliser comme une tension de repose fixe, de -70 millivolts à -30 millivolts (selon le neurone).

Lors de l'émission d'un influx nerveux, de petites portes moléculaires vont s'ouvrir dans la membrane, et vont laisser passer sélectivement certains ions, et pas les autres. Ainsi, on va trouver

des portes pour le potassium, d'autres pour le sodium, etc. Ces portes, des **canaux ioniques**, vont donc plus ou moins laisser passer un courant d'ions. Conséquence : ceux-ci sont modélisés par des résistances variables, qui laisseront plus ou moins bien passer le courant.

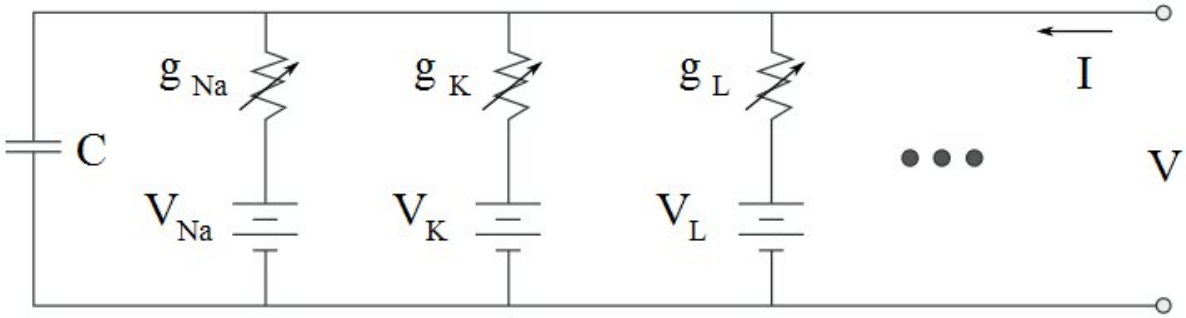
En supposant que seul un seul type d'ion puisse passer la membrane, on trouve alors un condensateur, et une seule résistance.

Pour résumer, voici quel est le modèle électrique du neurone choisi :



-> <-

Mais si on suppose que plusieurs types d'ions peuvent passer la membrane, alors on trouve un modèle plus complexe. Si les ions choisis sont le potassium, le sodium, et le calcium, on trouve un modèle de type Hodgin-huxley.



->
<-

Dans les deux cas, à partir de ces deux modèles de neurones, on peut obtenir une relation entre courant i et tension V qui a la forme d'une équation différentielle.

4.1 Circuits analogiques à base de transistors, résistances, et condensateurs

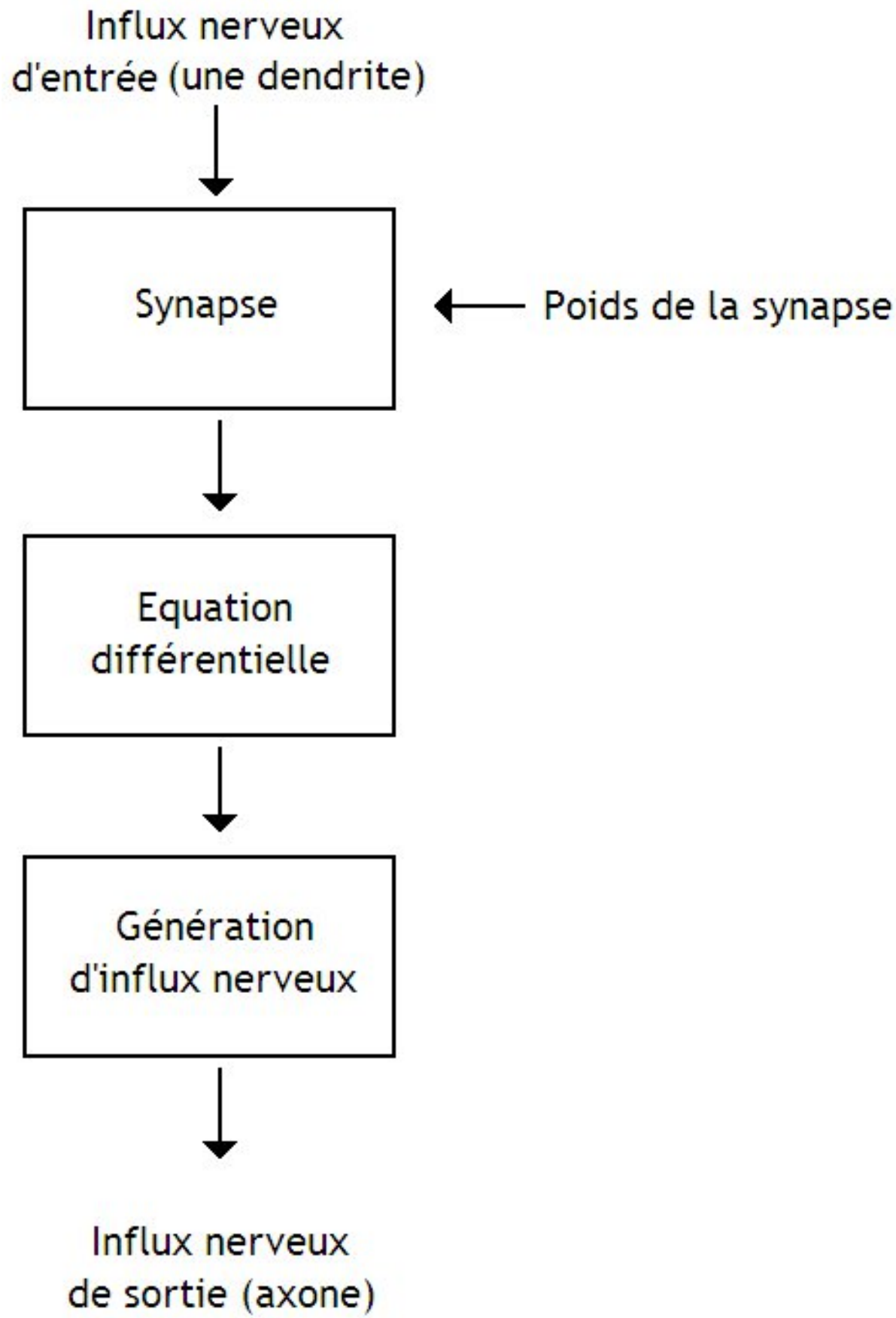
Les premières implémentations de neurones se basent sur des circuits analogiques. Cela vient du fait que dans les modèles de neurones, la tension évolue :

- soit via une équation différentielle ;
- soit, un potentiel d'action a lieu, faisant passer celle-ci à son maximum.

Or, simuler convenablement en matériel l'équation différentielle ne peut se faire convenablement avec des circuits numériques. Quoique, certains ont déjà essayés, avec parfois de bons résultats. Mais il est communément admis que l'analogique est roi pour ce genre de situations.

Quoiqu'il en soit, on trouve grosso-modo trois morceaux de circuit dans la majorité des implémentations matérielles de neurones :

- un circuit qui simule "le poids de la synapse" ;
- un circuit "mémoire", qui stocke l'état actuel de la tension, en faisant varier celle-ci par l'équation différentielle du modèle.
- et un circuit qui détecte le dépassement du seuil, et émet un influx nerveux.



->
<-

Il y a aussi quelques subtilités, mais on va simplement parler de l’essentiel.

Le circuit qui simule le poids de la synapse peut se résumer dans le cas le plus simple à un simple multiplieur analogique. Mais la majorité des design créent un générateur d'impulsions, des fronts de tension dont la durée dépend de la tension d'entrée (l'influx nerveux).

Le circuit qui implémente l'équation différentielle de la membrane en dehors d'un potentiel d'action est conçu à base de résistances, de condensateurs, et éventuellement de bobines : n'importe quel électronicien vous dira que toute équation différentielle peut être modélisée par un circuit ne contenant que des sources de tension, des condensateurs, des bobines, et des résistances.

Quant au circuit de détection du dépassement de seuil et de génération du nouvel influx nerveux, un simple amplificateur opérationnel correctement configuré peut suffire. Mais il faut signaler que chaque influx nerveux doit remettre la tension du neurone à la valeur de repos, ce qui demande quelques modifications sur le circuit de l'équation différentielle (un simple interrupteur bien placé peut suffire avec certains designs).

Pour en savoir plus sur ces design, vous pouvez lire les articles de recherche suivant, qui montrent quelques designs dans le genre :

- Hardware spiking neurons design : analog or digital ?
- Conception d'un neurone analogique Leaky Integrate-and-Fire - Cnfm
- Building blocks for electronic spiking neural networks

4.2 Memristors

Une solution plus économique est en cours de développement : le **Memristor**. Ce composant, inventé récemment, il y a quelques années, a un comportement proche d'une synapse. Le memristor n'est pas un composant électronique bien précis, et divers modèles existent a ce jour, chacun d'entre eux ayant des comportement très différents.

Ceux-ci peuvent être vus comme des résistances variables, avec quelques particularités. En conséquence, cette résistance a été renommée memristance, pour éviter les confusion.

Si jamais du courant positif traverse un memristor, sa memristance augmente. Si le courant est négatif, elle diminue. En clair : ces composants ont une sorte de mémoire du courant qui les a traversés depuis leur initialisation. Pour rentrer dans les détails, la valeur de la memristance dépend de l'intégrale du courant qui est passé dans la memristance. Suivant les modèles de memristor, cette fonction varie énormément.

Ces memristors permettent de modéliser fidèlement une synapse : il suffit de faire en sorte que les courants qui simulent un influx nerveux soient relativement courts. Grosso-modo, la memristance correspond au poids de la synapse.

5 Architectures AER

Les architectures neuromorphiques sont juste constituée d'un paquet de neurones et de synapses. Mais elles ne sont pas toutes organisées de la même manière. Le plus souvent, elles sont de deux types :

- les architectures AER ;
- les architectures totalement câblées.

Les architectures totalement câblées sont juste un amas de neurones et de synapses reliés entre eux par des synapses.

Mais pour les architectures AER, c'est autre chose : celles-ci sont composées d'une seule et unique synapse, multiplexée pour tous les neurones. Cette synapse va remplacer M synapses, grâce au multiplexage. Si la puce fonctionne à haute fréquence, cela ne pose pas de problème. Au pire, on traite les influx nerveux simultanés un par un.

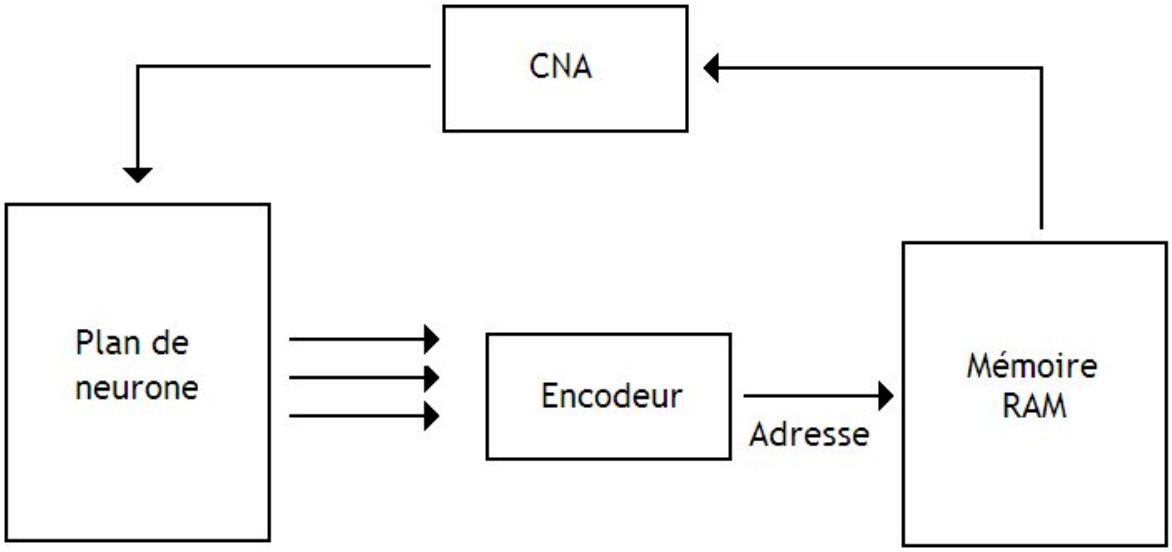
Évidemment, qui dit simuler plusieurs synapses avec une seule pose un problème : il faut ajouter le fait que chaque synapse a son propre poids. Il faut donc ajouter de quoi retenir le poids des synapses individuelles que l'on simule avec la synapse unique. Cela demande d'ajouter une mémoire RAM, qui stocke des nombres entiers qui simulent les poids.

Ces architectures sont composées des éléments suivants :

- un plan 2D ou 1D, voire 3D ; de neurones ;
- un décodeur ;
- un encodeur ;
- une synapse ;
- une mémoire pour stocker les différents poids ;
- un convertisseur numérique-analogique.

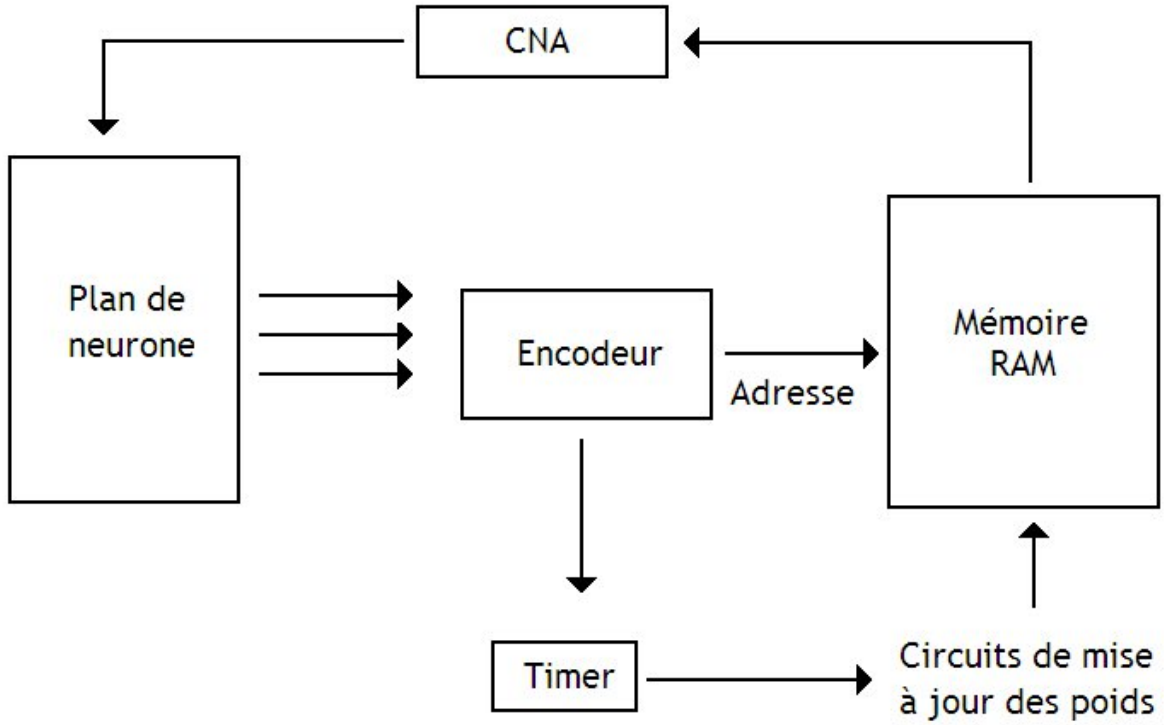
Le fonctionnement de ces architecture est simple. Chacun des neurones est numéroté : pour simplifier, on peut dire qu'il a une adresse.

Quand un neurone décharge et émet un influx nerveux, il faut déterminer son adresse : c'est le rôle de l'encodeur. Une fois son adresse connue, le poids de la synapse est récupéré dans la RAM, et est traduit en tension analogique par le convertisseur numérique-analogique. Ensuite, les neurones de destination sont adressés uns par uns, la tension précédemment convertie leur étant envoyée en entrée.



->
<-

Pour gérer le Spike Timing Dependant Plasticity, on peut ajouter des mémoires de type FIFO qui se chargent de mémoriser les temps d'émission des différents influx nerveux, et de calculer les mises à jour des poids dans la mémoire RAM en conséquence.



->
<-

Reste qu'utiliser un seul gros plan de neurones, cela bouffe du circuit, et qu'il faut tirer des fils à travers toute la puce. Je vous raconte pas les problèmes que cela pose, surtout quand on veut un chip qui fonctionne à haute fréquence. Pour limiter la casse, certains chercheurs ont décidé de casser ce plan de neurones en sous-plans de neurones, reliés entre eux par des réseaux locaux.

*[AER] : Adress Event Representation