Architectures neuronales : les réseaux de neurones comme architectures numériques

Neural architectures: neural networks as digital architectures

Bernard Girau - Biscuit team - Loria - Université de Lorraine

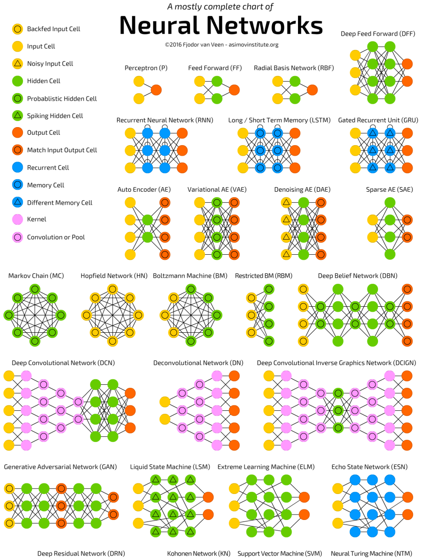
ARCHI'2017 - 10 mars 2017

Goal of this talk

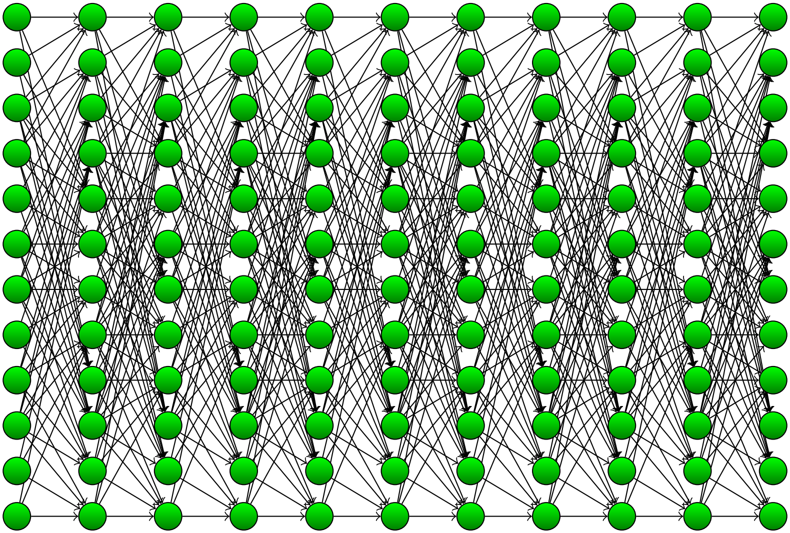
- Why a neural network may stand as a hardware architecture.
- Which kind of neural computation.
- Where it comes from (biological inspiration).
- Why it is not so simple to map neural networks onto digital hardware.
- How neural spikes partially solve the problems.

Artificial neural networks ?

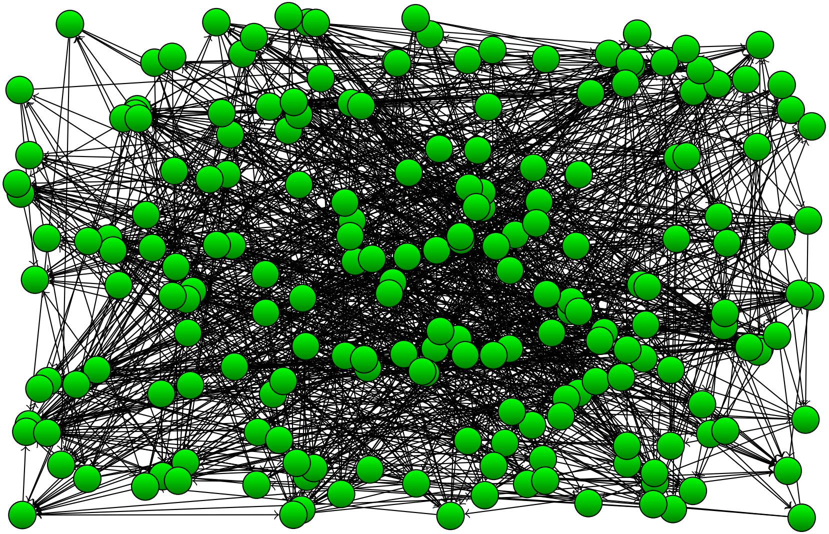
- several definitions
- many "architectures"
- graphs of small computing units that exchange data


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Example: convolutional network, n=121, c=630

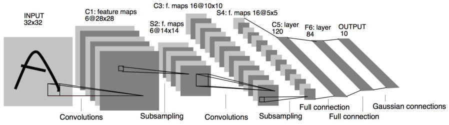


Example: liquid state machine, n=200, c=1200



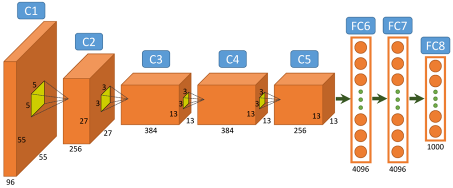
Some neural network sizes

- LeNet5 (1998) : $n = 8\,094,\ c = 582\,824$



©Y. Lecun

- AlexNet (2012) : $n = 594\,376,\ c = 947\,985\,976$



©F. Hu, G.S. Xia, H. Jingwen and Z. Liangpei

- Visual attention DNF model : $n = 9\,801,\ c = 36\,350\,000$

Context

- need to implement neural-based solutions on hardware devices
  - embedded system
  - speed up NN computation for statistical study
- search for cheap and flexible solutions (FPGAs ?)

Neural parallelism

- Neural networks are “naturally” parallel . . . not so simple !
- Different levels of neural parallelism, e.g. for standard feedforward NN:
  - session parallelism (mostly for learning)
  - data parallelism
  - **layer parallelism** (and thus pipeline)
  - **neuron parallelism**
  - **connection parallelism**
- About on-chip learning: only in specific conditions
  - to speed up learning of huge networks
  - to continuously adapt embedded system (e.g. ambulatory systems)

Hardware devices for neural networks

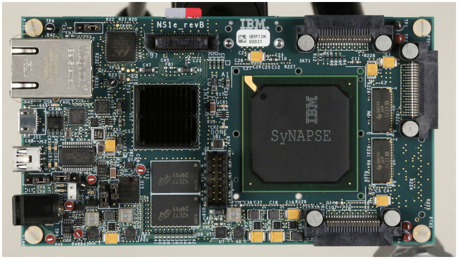
- Choice of parallel substratum: neural computation is fine-grain and requires dense interconnections.
- Hardware parallelism better fits specific aspects of neural parallelism.
- Analog hardware: yes, but does not fit the context.
- Several regular neural architectures fit GPU computations (convolutions, . . .): not presented here.
- Neuromorphic chips: not so accessible.

About neuromorphic chips . . .

- “old” approach: neuroprocessors, neuro-computers
- recent and booming trend: neuromorphic chips
- the ancestor ZISC (zero instruction set computer, 1993): 36 neurons
- the Cognitive Memory chip C1MK (2007): 1024 neurons, 0.5 mW

About neuromorphic chips . . .

- IBM SyNAPSE/TrueNorth (2014): 1 million neurons, 256 million synapses, 70 mW, 46 billion synapses computed per second and per watt, 5.4 billion transistors



©IBM

About neuromorphic chips . . .

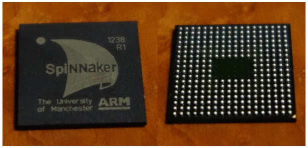
- Qualcomm Zeroth (2013-2015): now dedicated to deep learning in mobile solutions

©Qualcomm

- in the race: Intel, HP, Samsung

About neuromorphic chips . . .

- SpiNNaker (Human Brain Project, 2005-2014): 18000 neurons per chip, 500000 chip manycore architecture, flexible address-event connectivity



©Univ. Manchester

About neuromorphic chips . . .

- still difficult to use/access
- at the origin of this booming: spikes !

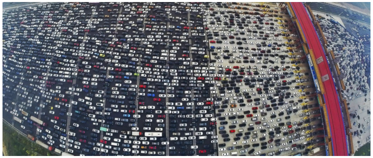
(back) Hardware devices for neural networks

- FPGAs: flexible, accessible, constantly improving
- Straightforward approach: directly map the neural architecture onto the chip
- Neurons: computing units, "operators"
- Connections: wiring

Implementation issues

NN may define their hardware architecture, but not so easy to map onto digital hardware devices ...

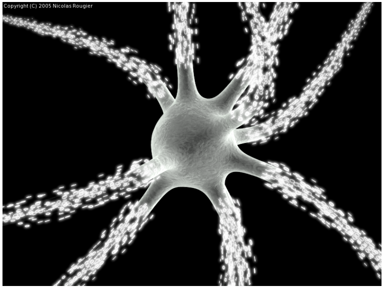
- Bandwidth issues
- Connection issues
- Area-greedy operators

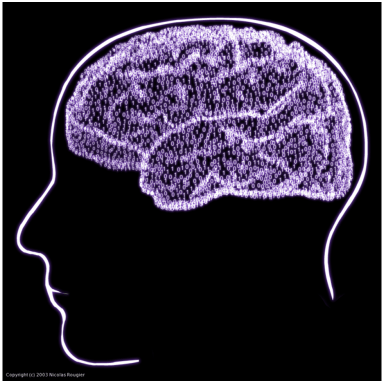


Implementation issues

Solving these issues requires to know more about neural networks.

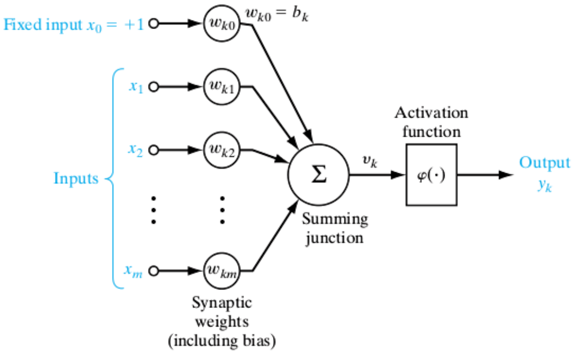
Neural modeling





Neuron ?

- Neuron models range from biologically plausible models (e.g. Hodgkin-Huxley type) to simplistic models (e.g. ReLU).
- Many neural networks use simple models like the McCulloch&Pitts neuron.



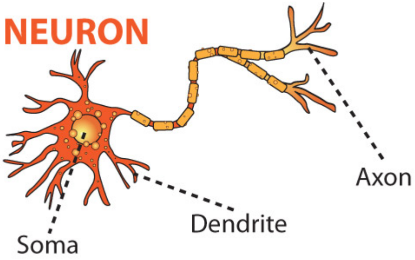
Neuron ?

but. . .

- Recent conceptual advances use more bio-inspired neurons.
- Even according to deep learning founders (LeCun, Bengio and Hinton), unsupervised and bio-inspired learning is “the future of deep learning”.

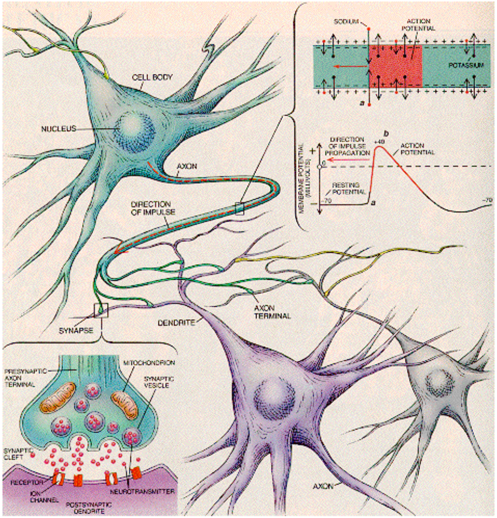
Back to biological foundations

- $\simeq 10^{11}$ neurons in the brain
- $\simeq 10^{15}$ dendrites
- Cell body: $\simeq 10\,\mu m$
- Axon: drives the neural signal (1 mm to 1 m) then branches
- Synapses: connect axon branches to dendrites of other neurons. Transmission of electrical signals between cells thanks to chemical processes.



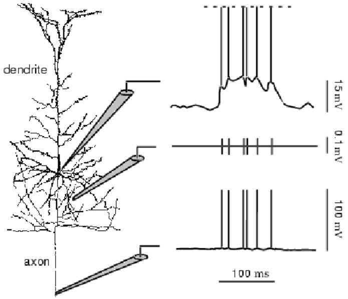
Back to biological foundations

- Membrane potential
- Ionic channels and pumps
- Resting potential
- **Action potential** (≃spike)
- **Post-synaptic potential** (PSP): inhibitory (IPSP), excitatory (EPSP)
- Neurotransmitters: bind to neuroreceptors to open channels.



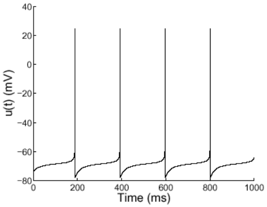
Back to biological foundations

- Accumulation of potential variations (received from dendrites) in the soma
- Non-linear processing: if the accumulated potential reaches a threshold, an action potential is generated at the basis of the axon
- Refractory period: unability to immediately generate new spikes

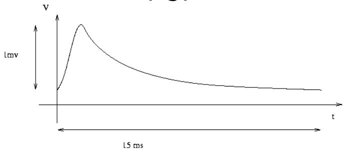


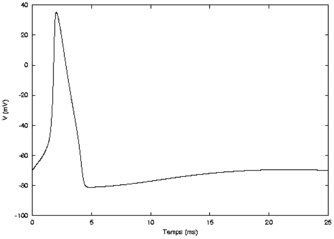
Back to biological foundations

Pulse/spike train



PSP

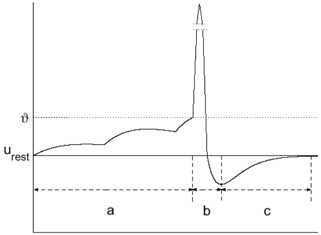


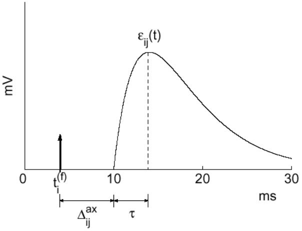


AP

Neuron modeling

Neuron models depend on how precise each component of the biological neural computation is modeled.





Spiking models (1/3)

- (biological modeling) electric and ionic mechanisms: biophysical models, e.g. Hodgkin-Huxley

$$\frac{dV_m(t)}{dt} = -\frac{1}{C_m}\left(I_{injected} + \sum_{ion} I_{ion}(t)\right)$$

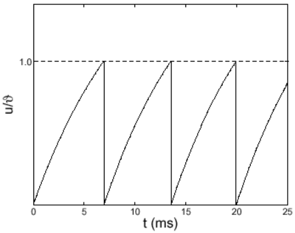
$$I_{ion}(t) = G_{ion} * m^p(t) * h^q(t) * (V_m(t) - E_{ion}(t))$$

- PSP and AP: e.g. SRM (spike response models)

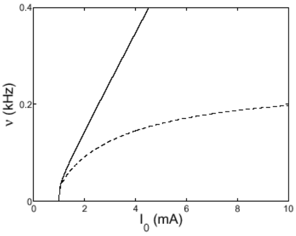
$$u_i(t) = \eta(t - t_i^{(f)}) + \sum_j w_{ij} \sum_{t_j^{(f)}} \epsilon_{ij}\left(t - t_i^{(f)}, t - t_j^{(f)}\right) + \int_0^\infty \kappa(t - t_i^{(f)}, s) I(t-s) ds$$

Spiking models (2/3)

- simple input integration and firing: e.g. IF and **LIF** (leaky integrate and fire)
  - membrane potential u: $\tau \frac{du}{dt} = -u(t) + \alpha I(t)$
  - firing: if $u(t) \geq \theta$ and $u'(t) \geq 0$, the neuron fires, firing time $t^{(f)} = t$
  - reset: $u(t^{(f)}) = 0$



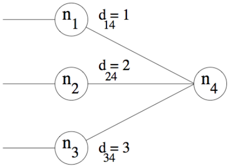
potential

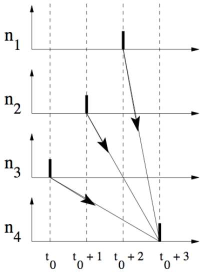


firing rate

Spiking models (3/3)

- with other neurons
  membrane potential: $\tau \frac{du_i}{dt} = -u_i(t) + \alpha I_i(t) + \sum_j w_{ij}\delta_j(t)$
- firing: $\delta_i(t) = \mathrm{Dirac}(t - t^{(f)})$
- with delays: $\tau \frac{du_i}{dt} = -u_i(t) + \alpha I_i(t) + \sum_j w_{ij}\delta_j(t - d_{ij})$





Rate-coded models

- average activity: e.g. formal neuron, ReLU

$$y_i = \phi\left(b_i + \sum_j w_{ij} x_j\right)$$

- with time: e.g. recurrent networks

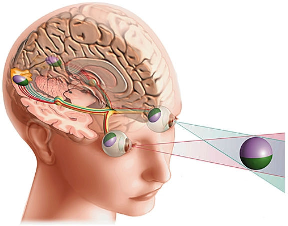
$$y_i(t+1) = \phi\left(b_i + \sum_j w_{ij} x_j + \sum_k w_{ik} y_k(t)\right)$$

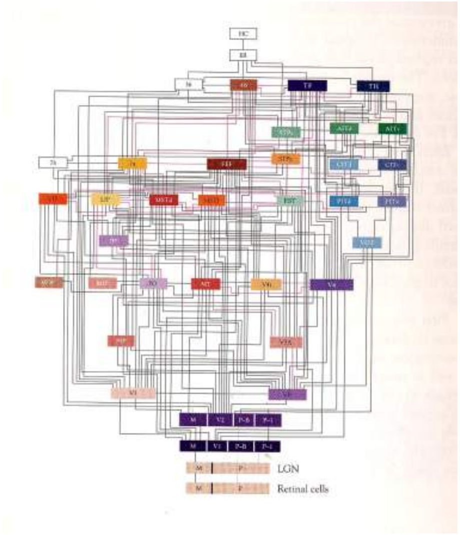
- dynamic activity: e.g. **DNF**, dynamic neural fields

$$\tau \frac{du_i}{dt} = -u_i(t) + \alpha I_i(t) + \sum_j w_{ij} f(u_j(t)) + h$$

Neural population dynamics

- Population coding
- Emergent computation
- Temporal computing
- Inspiration: lateral and feedback connections in the brain, e.g. visual system





Elementary dynamic neuron

Let's choose a simple rate-coded neuron model.

$$u'_t = \frac{1}{\tau}(-u_t + I_t + h)$$

- leak, input, resting potential
- continuous integration of informations

Implementation

- discretization ($u' = f(t,u)$):

$$u_{t+dt} = u_t + slope(t, u_t, dt)$$

- numerical solutions of differential equations: multistep methods, Runge-Kutta methods ?
- Biological modeling: RK4

$$\begin{aligned} slope(t, u_t, dt) &= \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \\ & k_1 = f(t, u_t) \quad k_2 = f(t + \frac{dt}{2}, u + k_1\frac{dt}{2}) \\ & k_3 = f(t + \frac{dt}{2}, u + k_2\frac{dt}{2}) \quad k_4 = f(tdt, u + k_3 dt) \end{aligned}$$

Implementation

- Bio-inspired computing: first order methods appear sufficient

$$\text{(Euler)} \qquad u_{t+dt} = u_t + \frac{dt}{\tau}(-u_t + I_t + h)$$

justifications: “neurons are robust”, “small order, small dt”, “simpler implementation ”, “dynamic behaviour is maintained”, . . .

Maps of dynamic neurons

Let's make our neuron able to interact with other neurons

$$u'_t = \frac{1}{\tau}(-u_t + Lat_t + Aff_t + h)$$

- lateral information: from other neurons
- afferent information: from some "external" inputs

Let's also give a position x to our neuron

$$u'_{x,t} = \frac{1}{\tau}(-u_{x,t} + Lat_{x,t} + Aff_{x,t} + h)$$

Maps of dynamic neurons

Let's take into account synaptic influence

$$u'_{x,t} = \frac{1}{\tau}(-u_{x,t} + \int_{x'} w(x,x')f(u_{x',t}) + Aff_t + h)$$

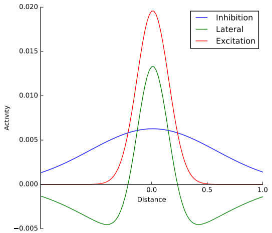
- synaptic weights
- integration over the whole population

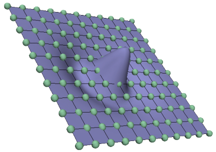
Example: dynamic neural fields (DNF)

If *w* only depends on the inter-neuron distance, e.g. according to some difference of gaussians ... we obtain standard DNFs

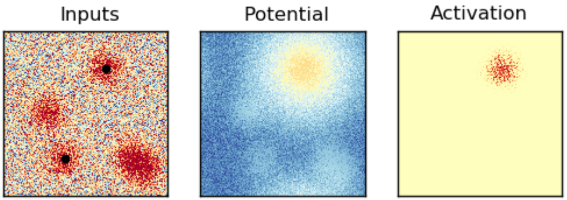
$$w(x, x') = \omega(||x - x'||)$$

$$\omega(d) = Ae^{\frac{-d^2}{a^2}} - Be^{\frac{-d^2}{b^2}}, A, a, B, b \in \mathbb{R}_+^*.$$





DNF: dynamic neural fields

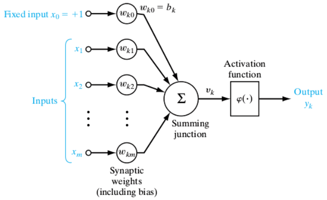


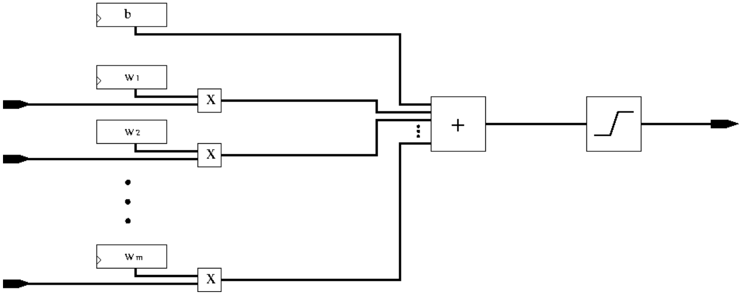
- Each map capable of target selection, tracking, etc.
- Numerous higher level applications: visual attention, motor planning, change detection, etc.
- Decentralization and robustness

Demo ???

Simple formal neuron

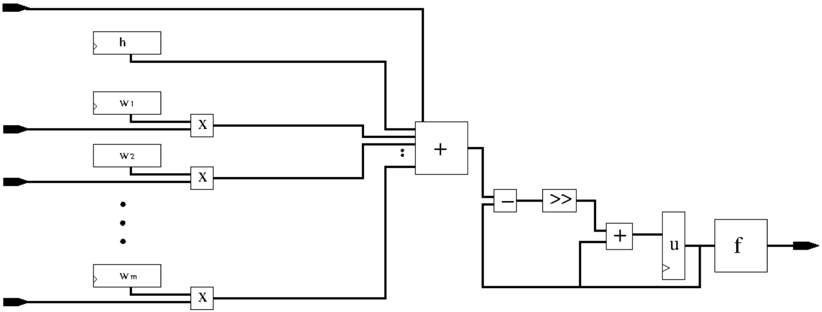
$$y_i = \phi\left(b_i + \sum_j w_{ij} x_j\right)$$





Dynamic neuron

$$\tau \frac{du_i}{dt} = -u_i(t) + \alpha I_i(t) + \sum_j w_{ij} f(u_j(t)) + h$$



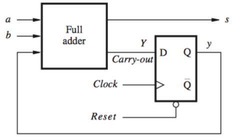
Connecting neurons

- The network is the architecture.
- Unsolved issues
  - bandwidth
  - dense interconnections
  - **area-greedy operators**

Arithmetic tricks

To obtain smaller operators:

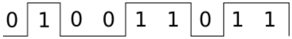
- serial arithmetic (storing weights in LUTs)



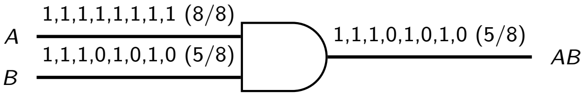
- pipeline between operators: non-linear units are MSBF
- on-line arithmetic: serial & MSBF

Arithmetic tricks

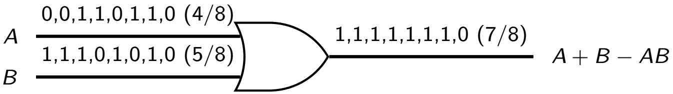
- bitstream arithmetic
  - Each real value is encoded in a stream of bits
  - Encoding of value $x \in [0, 1]$ uses a bitstream where for each bit b, $P(b = 1) = x$



  - Compact multiplication: $P(A \wedge B) = P(A)P(B)$



  - Biased addition: $P(A|B) = P(A) + P(B) - P(A \wedge B)$



Arithmetic tricks

- advantages of bitstream arithmetic
  - compactness
  - anytime computation
  - biased addition stands for non-linearity
- limits of bitstream arithmetic
  - kinds of operators
  - precision (related to bitstream length)
  - correlated random variables (long computation paths in neural networks)
  - generation of random numbers

Spiking neuron

Neurons know how to deal with a constrained bandwidth.

- action potentials, post-synaptic potentials, etc.
- ok, but … Hodgkin-Huxley, Izhikevich, SRM, LIF, etc. ???
- computer scientists are “binary”

$$u_{x,t+dt} = \begin{cases} u_{x,t} + \frac{dt}{\tau}(-u_{x,t} + I_{x,t} + h) & \text{if } u_{x,t} < \theta, \\ h & \text{if } u_{x,t} \geq \theta \end{cases}$$

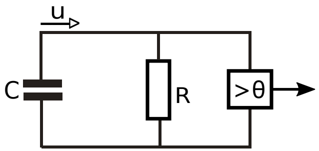


Figure: LIF neuron as RC circuit

Spiking neural populations

Same architecture, but neurons only exchange “all or nothing” information.

$$Lat_{x,t} = \sum_{x'} w(x, x') S_{x',t} \tag{1}$$

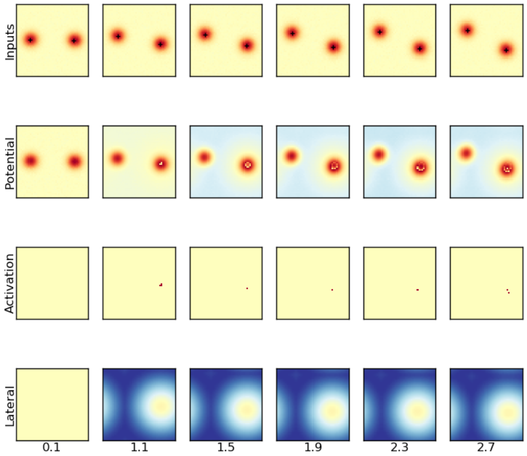
$$S_{x,t} = \begin{cases} 1 & \text{if } u_{x,t} \geq \theta \\ 0 & \text{if } u_{x,t} < \theta. \end{cases} \tag{2}$$

This is so . . . bandwidth-friendly. But does it work ?

Current applications of spikes

- applications in perception
- liquid state machines
- spiking deep learning
- spiking neural fields
- ...

Example: spiking dynamic neural fields



Spikes on-chip

Spike-based computation reduces bandwidth demands.

It also reduces the implementation area of each neuron:

- slightly more complex neuron (thresholding), but ...
- no more multiplier

$$Lat_{x,t} = \sum_{S(x',t)=1} w(x,x')$$

- multiplication by $\frac{dt}{\tau}$: if dt is small enough to ensure enough accuracy, reduce it further until $\frac{dt}{\tau}$ is 2^{-p} and use simple binary shifts.

Spikes on-chip

Receiving spikes:

- temporal discretization: spikes are events, yet they appear in the differential equations
  - with the LIF model, spikes are instantaneous

$$u_{x,t+dt} = \begin{cases} u_{x,t} + \frac{dt}{\tau}(-u_{x,t} + Aff_{x,t} + h) + \frac{1}{\tau}\sum_{S(x',t)=1} w(x,x') & \text{if } u_{x,t} < \theta, \\ 0 & \text{if } u_{x,t} \geq \theta \end{cases}$$

  - with a more detailed PSP model, back to numerical simulation of differential equations
- multiplication of weights by $\frac{1}{\tau}$: just adapt weights

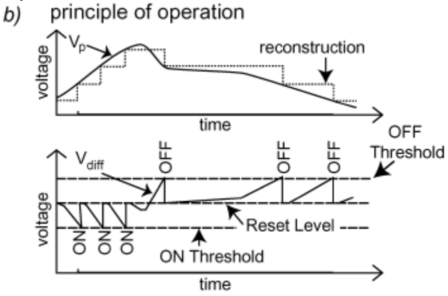
Spikes on chip

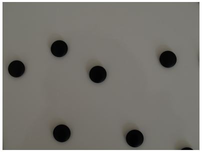
Communicating spikes:

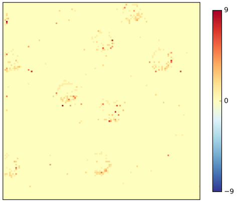
- simpler handling of dense interconnections (see further)
- basic idea: one (or just several) spike at a time
- only meaningful information to be sent
- towards asynchronous implementations:
  - each spike is a local clock event
  - no global clock in IBM TrueNorth

An example

- Input: DVS sensor

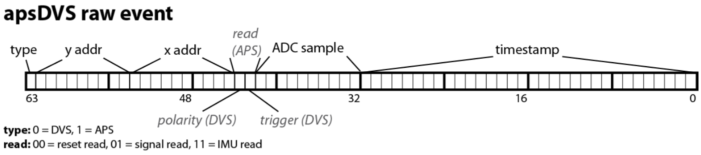






An example

- spike transmission: AER protocol/bus (address-event representation)
  - different formats
  - spike type
  - location information
  - time stamp
  - etc.

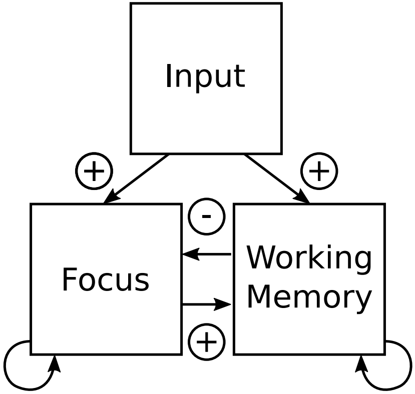


©IniLabs

- may be used between chips or within chips

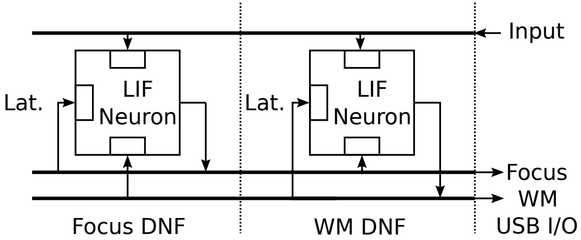
An example

- neural model: DNF-based



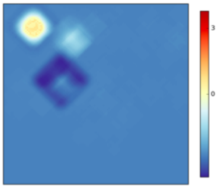
An example

- architecture

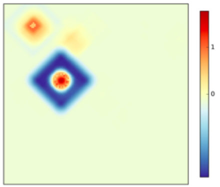


- 10000-neuron DNF (10^8 connections) on a single FPGA

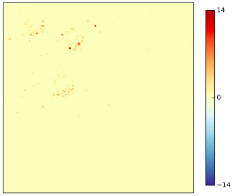
An example



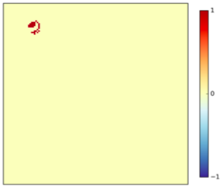
(b) u Focus



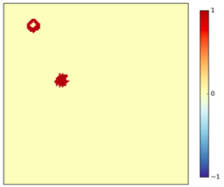
(c) u WM



(a) DVS



(d) act. Focus



(e) act. WM

- Neural parallelism and hardware parallelism: soon reconciled ?
- Spikes not limited to modeling biology
- Spikes make drastic computation simplifications possible for digital hardware.
- Spikes make the world a bit more . . . binary.
- Many other spiking tricks:
  - randomly propagating spikes
  - spike-stream computation: robustness to correlated bitstreams thanks to potential reset
  - etc.

The end.