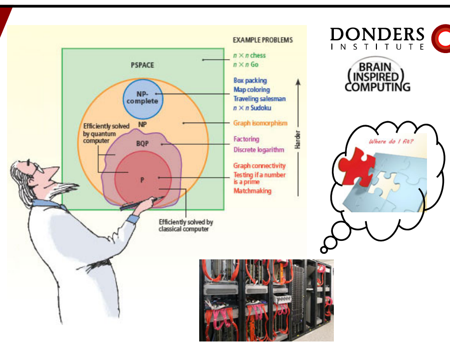


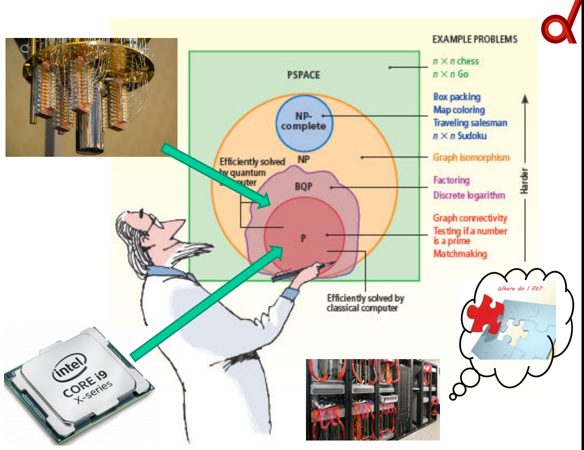
DONDERS INSTITUTE

BRAIN INSPIRED COMPUTING

Towards Neuromorphic Complexity Analysis

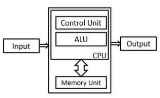
Johan Kwisthout, Associate PI, Donders Institute

Radboud University · Radboudumc



Brain-inspired computing

- In October 2015, the White House Office of Science and Technology Policy released the following Grand Challenge for Future Computing:
- *"Create a new type of computer that can proactively interpret and learn from data, solve unfamiliar problems using what it has learned, and operate with the energy efficiency of the human brain"*

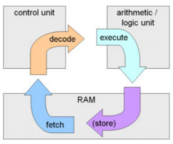


Von Neumann Model

Neural-inspired systems for ***non*-von Neumann** computational architectures

Continuous data transfer between memory and CPU

Heat generation bottleneck!



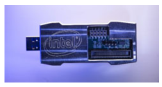
Neuromorphic architectures

What *is* a neuromorphic architecture?

- Principles most people agree on
  - Neurons and Synapses as basic components
  - Co-located memory and computation
  - Potentially energy efficient ("orders of magnitude")
- Different schools of thought
  1) Analog hardware based ("memristors")
  2) Spike-based digital systems ("spiking neural networks")

BrainScaleS

SpiNNaker



Loihi

Neuromorphic architectures

What are these neuromorphic architecture good for?

- Probably well suited for…
  - Event-driven sensors / actuators (neuromorphic robotics)
  - Energy-critical applications that allow for less precision
- Probably not the best architecture for…
  - 'Deep' classification and pattern recognition (outperformed by convolutional DNNs)
  - Applications that value precision over energy usage
- Is "machine learning" the only possible application?

Remember the GPU…

Programming neuromorphic architectures

- Some problems (e.g., matrix multiplication) can be solved more efficiently on a GPU; for others (e.g., serial computations) a CPU is still better suited
- Programming a GPGPU requires different way of thinking about data structures (e.g. data streams)
- Similarly, some problems may be solved more (energy-efficiently) on neuromorphic architectures than on more traditional Von Neumann architectures
- Similarly this requires different ways of thinking about data and algorithms!

Towards neuromorphic complexity analysis

- What sort of problems are efficiently solvable on a neuromorphic computer? Which are not? Are these problems different / the same as the problems efficiently solvable on a Von Neumann architecture?
- Given the nature of neuromorphic architectures, **energy** seems to be a vital resource (not only time)
- Our current models of computation (viz., Turing machines) capture only time and space as relevant resources for computation – not energy!

New computational model is needed

- DoE 2016 workshop report, p. 29:
  *"…likely that an **entirely new computational theory paradigm** will need to be defined in order to encompass the computational abilities of neuromorphic systems"*

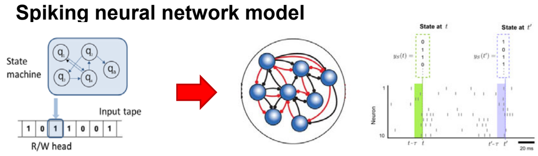
Goal: To describe what sort of **problems** can and cannot be solved **energy-efficiently** on neuromorphic hardware

Needed: New branch of complexity theory with:

1) Formal notion of "computation" in neuromorphic architectures
2) Complexity classes based on resource constraints
3) Hardness criteria and a means to *translate* problems into each other while keeping resources invariant
4) Algorithms to show that a problem is in a specific class

Proposed computational framework

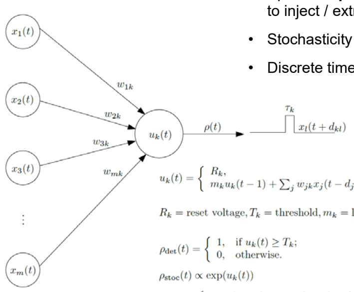
Spiking neural network model



- Key neuromorphic aspects are there:
  - Co-located memory & computation
  - Spiking behavior → energy efficiency
  - Stochastic sampling or deterministic spikes
- Underlying principle of Loihi / SpiNNaker machines

Neuronal model: basically simple LIF model

- Specific **input** and **readout** neurons to inject / extract information
- Stochasticity or determinism
- Discrete time steps



$$u_k(t) = \begin{cases} R_k, & \text{if } u_k(t-1) \geq T_k; \\ m_k u_k(t-1) + \sum_j w_{jk} x_j(t - d_{jk}), & \text{otherwise.} \end{cases}$$

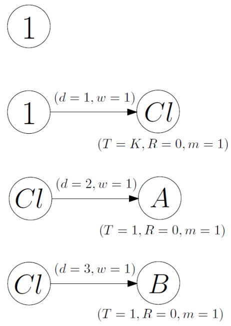
R_k = reset voltage, T_k = threshold, m_k = leakage constant

$$\rho_{\text{det}}(t) = \begin{cases} 1, & \text{if } u_k(t) \geq T_k; \\ 0, & \text{otherwise.} \end{cases}$$

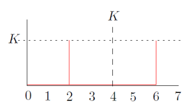
$$\rho_{\text{stoc}}(t) \propto \exp(u_k(t))$$

$$x_k(t) = \begin{cases} 1, & \text{if a spike was released in } \langle t - \tau_k, t]; \\ 0, & \text{otherwise.} \end{cases}$$

Example problem: adding two numbers mod *K*

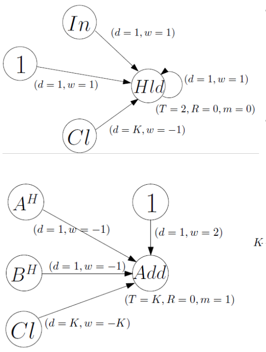


- Special neuron Const fires at every time step >= 0 (bias = threshold)
- Clock neuron fires every *K* timesteps
- A and B fire at time *d* mod *K (d = 2,3)*

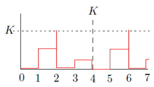


- Spiking time of the neuron relative to clock encodes its value!

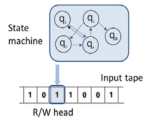
Example problem: adding two numbers mod K



- Circuit ensuring that once a neuron fires, it fires until reset by the clock
- Adder circuit spikes on carry-out, neuron potential at clock intervals encodes A+B mod *K*

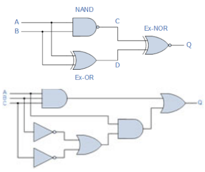


Beyond Turing



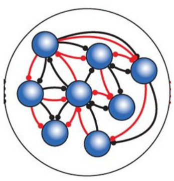
- **Turing Machine M_L**
- Input **I** encoded (in binary) on the tape
- State machine M_L implements algorithm
- Formally: recognizes languages $L \subset \{0,1\}^*$
- Canonical question: Does M_L accept $I \in L$ using resources (time/space) at most **R**?

- ***Family* of Boolean Circuits $C_{L,|I|}$**
- Input **I** encoded as special input gates
- Circuit (different circuit per input size |I|) implements algorithm
- Formally: recognizes languages $L \subset \{0,1\}^*$
- Canonical question: Does, for every I, the corresponding circuit $C_{L,|I|}$ accept $I \in L$ using resources (time/space) at most **R**?



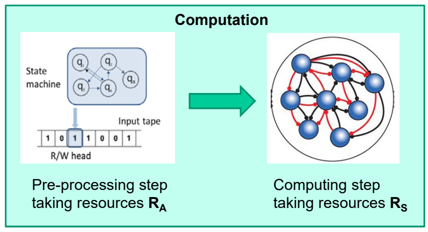
Beyond Turing

- In SNNs, input **I** and algorithm **A** are **co-located!**
- We take the circuit idea *to the extreme*…



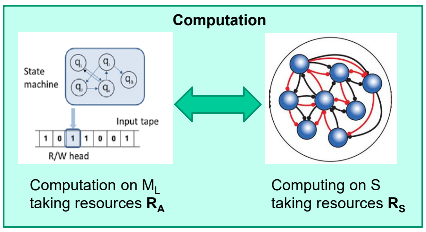
- ***Collection* of SNNs $S_{L,I}$**
- *One* network for *every* input **I** (or set of inputs {**I**})
- Input and algorithm operating on it are encoded in the network structure
- Formally: recognizes languages $L \subset \{0,1\}^*$
- Accept / reject by special neurons firing
- Canonical question: Is there a resource-bounded Turing machine M_L that, given **I**, generates $S_{L,I}$ which decides **I** using resources at most R_S?

Beyond Turing: preprocessing + computation



- **Hierarchy** of complexity classes defined by choices for R_A(time, space) and R_S(time, space, energy)
- E.g., R_A(poly time, log space), R_A(poly time, space, energy)

Computing with neuromorphic oracle



- More powerful alternative: use S as co-processor
- Formally: S is an *oracle* for Turing machine M_L

Some first theoretical results

- We have defined canonical complete problems and resource-preserving reductions between problems

SNN-Halting
Instance: Integers t, e in unary notation; encoding $\mathcal{S} = (N, S)$ of a spiking neural network constructable by a Turing machine $\mathcal{M}(R_A)$.
Question: Does N_{acc} fire before time step t using energy at most e?

Oracle SNN-Halting
Instance: Integers t, s in unary notation; encoding of $\mathcal{M}(R_A)^{\mathcal{S}(R_S)}$; input I.
Question: Does $\mathcal{M}(R_A)^{\mathcal{S}(R_S)}$ accept I before time step t, accessing at most s cells on its tape?

- These problems is complete for resource bounded SNNs, resp. neuromorphic oracle machines, and can be used as starting point for reductions to actual (real-world) problems

Some first theoretical results

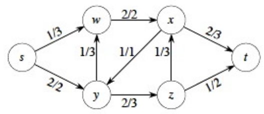
- **Neuromorphic oracles cannot do magic…**

$$P^{\mathrm{SNN}(\mathcal{O}(n^c),\mathcal{O}(n^c),\mathcal{O}(n^c))} = P$$

- This implies that there is no super-polynomial speedup in using a neuromorphic co-processor (which is similar as for using GPUs)
- But then again, speed is not the major consideration when we are considering neuromorphic hardware
- But **energy** is!

Some first theoretical results

Max network flow problem



This problem is P-complete, meaning that it cannot be efficiently parallellized (use only logarithmic space) on traditional machines!

MAX NETWORK FLOW is in $L^{\mathrm{SNN}(\mathcal{O}(n),\mathcal{O}(n),\mathcal{O}(n))}$

It can be solved in Logspace when allowed to use a neuromorphic co-processor!

MSc project Abdullahi Ali

MAX Network Flow

- Max Network Flow is P-complete, implying it is an inherently serial problem resisting effective parallelization – takes more than log(n) space
- We can *not* solve it efficiently on a 'stand-alone' neuromorphic computer (working on formal proof)
- Idea: use the power of the neuromorphic oracle
  - Offload those computations that can be done in parallel
  - Specifically: finding the shortest augmenting path between s and t using "first-past-the-post" spike timing
  - We can show that the CPU then needs only log(n) space and the oracle (for dense networks) log(n) spikes

Neuromorphic complexity in theory and practice

- Theoretical research financially supported by Intel's Neuromorphic Research Community
- Goal of INRC is to build a research community around their new Loihi chip
- We try to implement our algorithm not only in (abstract) simulator but in actual hardware
- Show the relation (or mismatch...) between formal theory and practice!

Neuromorphic complexity in theory and practice

- Theoretical model: **oracle state** / **oracle tape** for communication between TM and SNN
- Machine model allows for definition of **readout neurons** whose state can be written to the tape
- In reality, this doesn't work this way: you just "upload" the network and "download" the entire SNN state → computationally costly!
- We have not yet taken **constrained** communication resources into consideration

Conclusion

- New theoretical framework
  1) Formal notion of "computation" in neuromorphic architectures
  2) Complexity classes based on resource constraints
  3) Hardness criteria and a means to *translate* problems into each other while keeping resources invariant
  4) Algorithms to show that a problem is in a specific class
- Iteration between formal model and actual architecture to ensure vital aspects are captured
- My research group is currently working on GUI to develop and 'drag-and-drop' design patterns (circuits) to construct algorithms