On the Computational Power and Complexity of Spiking Neural Networks

Johan Kwisthout

johan.kwisthout@donders.ru.nl
https://www.socsci.ru.nl/johank/index.html/

Joint work with Nils Donselaar and Arne Diehl

Donders Center for Cognition
Department of Human-Centred Intelligent Systems

Workshop on Theory of Neuromorphic Computing - June 2026

Overview

- Neuromorphic computing: interesting new approach
  - Intel Loihi, Manchester Spinnaker systems
  - Energy as resource on par with time and space
  - Potential for orders of magnitude more efficient
  - But hard to compare with Von Neumann architectures
- Can we have a neuromorphic complexity theory?
- Is this architecture really something new?
- Are the inherently energy-inefficient problems?
- Invitation for collaborations!

Agenda

- Introduction
  - Who am I?
  - Research group

Agenda

- Introduction
  - Who am I?
  - Research group
- Motivation - why this work?
  - Neuromorphic computing
  - Previous work

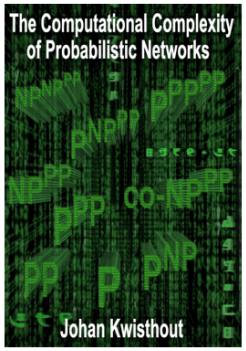
Agenda

- Introduction
  - Who am I?
  - Research group
- Motivation - why this work?
  - Neuromorphic computing
  - Previous work
- Machine model for neuromorphic computing

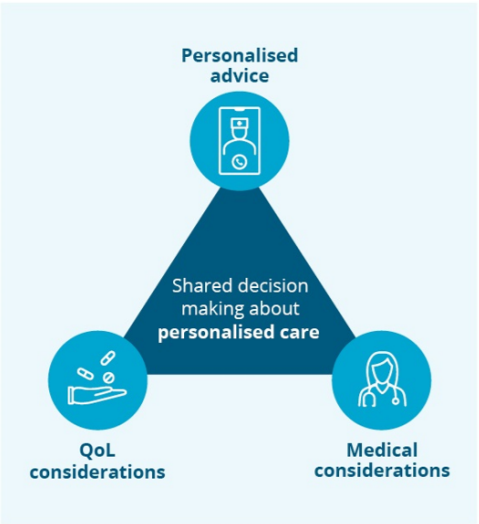
Agenda

- Introduction
  - Who am I?
  - Research group
- Motivation - why this work?
  - Neuromorphic computing
  - Previous work
- Machine model for neuromorphic computing
- Structural complexity results

Who am I



PersOn | Personalised Care in Oncology



ELSA lab
for Decision Support

P G M
G R O U P

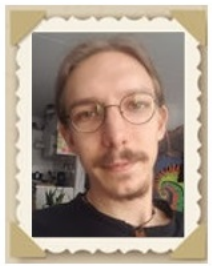
<AI>RADBOUD UNIVERSITY</AI>
Artificial Intelligence Program

Research Group

Research Group

- PGMs in Decision Support Systems
  - Explainability, trustworthiness, accountability, QoL, online maintenance, traceability
  - Foundations: Credal networks, Learning, Model checking
  - Applications: Healthcare, AutoML, Sustainability
- Foundations of stochastic computing
  - Parameterized complexity, approximate inference
  - Stochastic computing, synchronized stochasticity
  - Neuromorphic computing




Intel INRC

- I have been part of (and modestly supported by) Intel's Neuromorphic Research Community from its start in 2019
- Work presented today stems from this period

Intel INRC

- I have been part of (and modestly supported by) Intel's Neuromorphic Research Community from its start in 2019
- Work presented today stems from this period
- Our university was the second programme in the world (Carnegie Mellon being the first one) that offered students (remote) access to Intel's Loihi chip ...
- ... and we were among the first that offered a graduate course on neuromorphic computing
- However, I am currently not very active in the field anymore

Paper

- Most of today's presentation comes from this paper: `https://iopscience.iop.org/article/10.1088/2634-4386/ae2cc1`

Neuromorph. Comput. Eng. **6** (2026) 014004 https://doi.org/10.1088/2634-4386/ae2cc1

NEUROMORPHIC
Computing and Engineering

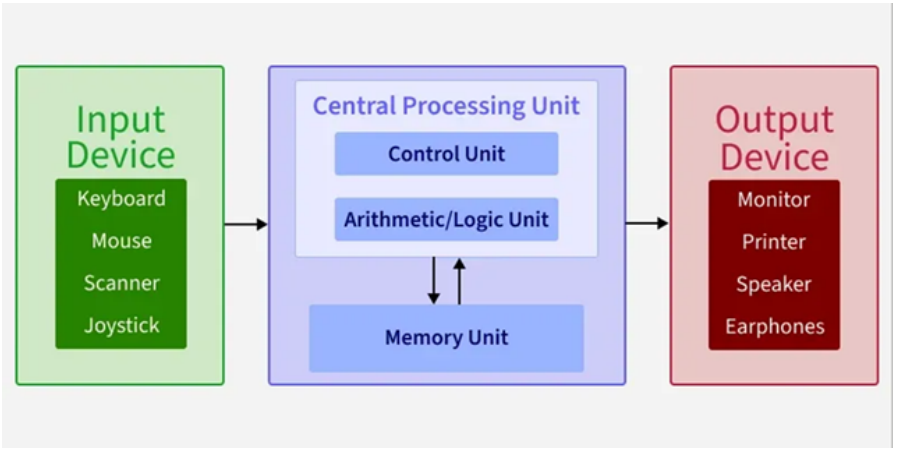
PAPER

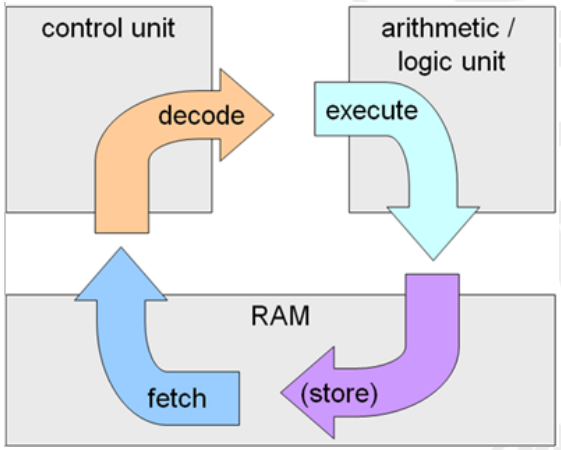
Neuromorphic complexity theory: computational models and complexity classes for spiking neural networks*

Johan Kwisthout[1], Arne Diehl[**] and Nils Donselaar[2]

Von Neumann architecture

- Let's start describing the Von Neumann architecture and processing cycle, just to explain what is different in neuromorphic computing





Motivation

- Limits to Von Neumann architectures
  - Moore's law ending: number of transistors limited
  - Carbon footprint vs Landauer limit
  - Bottleneck: transfer data & code from memory to CPU

Motivation

- Limits to Von Neumann architectures
  - Moore's law ending: number of transistors limited
  - Carbon footprint vs Landauer limit
  - Bottleneck: transfer data & code from memory to CPU
- What is neuromorphic computing?
  - Sort of container term for 'brain-inspired computing'
  - In this talk: architectures where data and algorithm are co-located in spiking neural network architectures

Motivation

- Limits to Von Neumann architectures
  - Moore's law ending: number of transistors limited
  - Carbon footprint vs Landauer limit
  - Bottleneck: transfer data & code from memory to CPU
- What is neuromorphic computing?
  - Sort of container term for 'brain-inspired computing'
  - In this talk: architectures where data and algorithm are co-located in spiking neural network architectures
- What can we do with these architectures?

Motivation

- Limits to Von Neumann architectures
  - Moore's law ending: number of transistors limited
  - Carbon footprint vs Landauer limit
  - Bottleneck: transfer data & code from memory to CPU
- What is neuromorphic computing?
  - Sort of container term for 'brain-inspired computing'
  - In this talk: architectures where data and algorithm are co-located in spiking neural network architectures
- What can we do with these architectures?
- What can they do better than CPUs / GPUs?

Motivation

- Remember the introduction of the GPU?
  - **Graphics** Processing Unit
  - Meant to speed up computations necessary for graphics (shadows, motion, etc.)
  - Effective large matrix computations

Motivation

- Remember the introduction of the GPU?
  - **Graphics** Processing Unit
  - Meant to speed up computations necessary for graphics (shadows, motion, etc.)
  - Effective large matrix computations
- Nobody thought of deep learning as useful application of GPUs prior to AlexNet crushed the competition in 2012...
  - Now 90% of GPU energy use is in AI...

Motivation

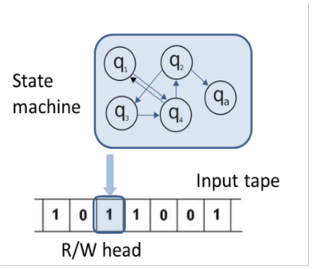
- Remember the introduction of the GPU?
  - **Graphics** Processing Unit
  - Meant to speed up computations necessary for graphics (shadows, motion, etc.)
  - Effective large matrix computations
- Nobody thought of deep learning as useful application of GPUs prior to AlexNet crushed the competition in 2012...
  - Now 90% of GPU energy use is in AI...
- Lesson learnt: Maybe there are other tasks that can be sped up with a neuromorphic chip besides machine learning?
- For example, scientific computing, graph-like problems

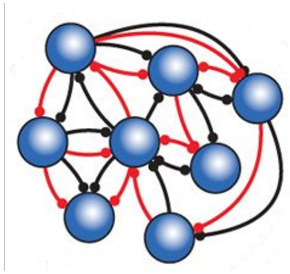
Example problems with neuromorphic algorithms

- Finding shortest paths (Dijkstra's algorithm)
- Finding minimum spanning trees
- Finding number of dominating sets
- Longest increasing subsequence
- We worked on several of these problems and built a GUI / simulator for hand-crafting networks: `https://github.com/nielsvharten/SNN-builder/`

Spiking Neural Network (SNN)

- Spiking Neural Network model as computational model for neuromorphic architectures



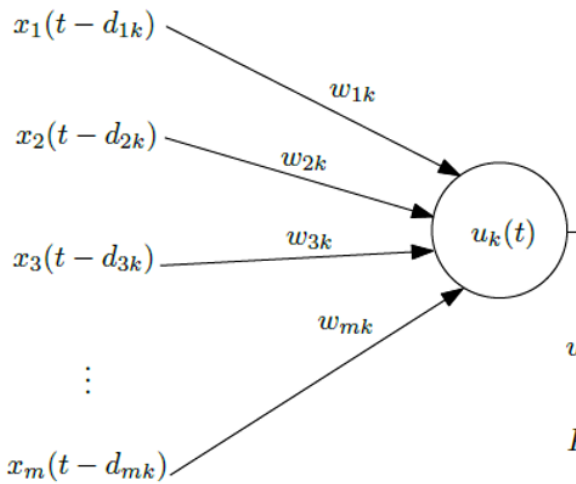


- Deterministic neurons
- Specific I/O neurons
- Discrete time steps

- Key neuromorphic aspects are there:
  - Co-located memory & computation
  - Spiking behaviour $\rightarrow$ energy efficiency
  - Underlying principle of Loihi / SpiNNaker machines
- Abstraction – some properties are lost

SNN definitions

- We have programmed input neurons, dedicated readout neurons, & regular LIF neurons with potentials, threshold and reset voltages, and leakage; synapses with delays and weights

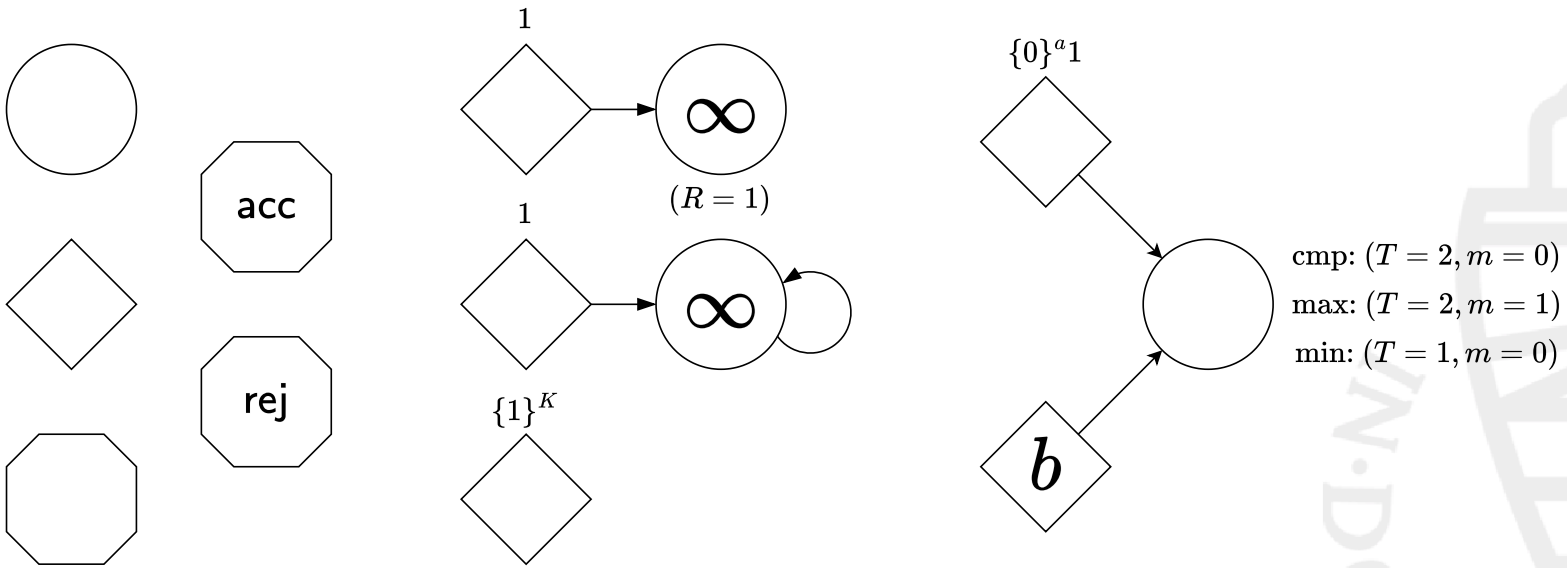


$$u_k(t) = \begin{cases} \max(0, R_k + \sum_j w_{jk} x_j(t-d_{jk})) & \text{if } u_k(t-1) \geq T_k; \\ \max(0, m_k u_k(t-1) + \sum_j w_{jk} x_j(t-d_{jk})) & \text{otherwise.} \end{cases}$$

$R_k = \text{reset voltage}, T_k = \text{threshold}, m_k = \text{leakage constant}$

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

Notation and simple circuits



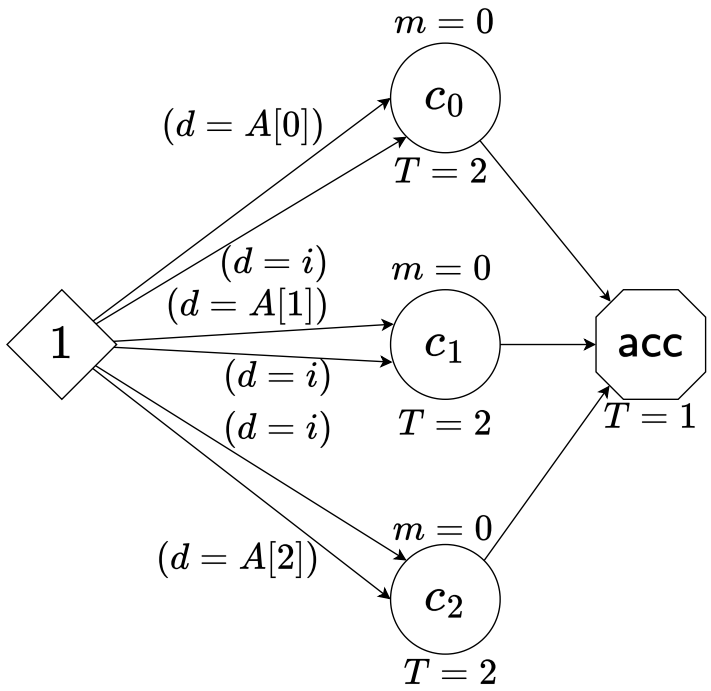
Left: regular, input, and readout neuron; dedicated accept, reject neuron

Mid: three ways of encoding a constantly firing neuron

Right: comparing a natural number against a reference value

Solving a simple problem: take 1

Decide whether an array A of natural numbers contains i.

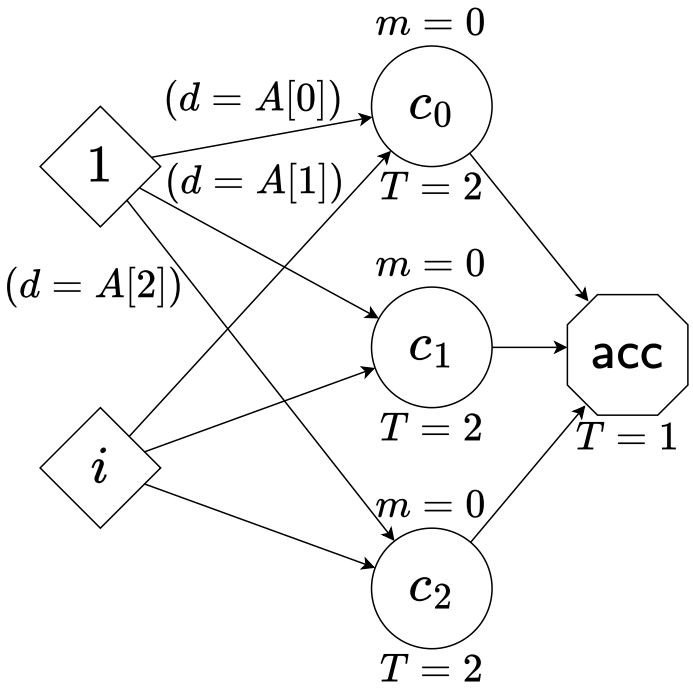


Concept: both A and i (the data) as well as the parallel comparison (the computation) are encoded in the machine

Energy: $\leq |n| + 2$ spikes needed

Solving a simple problem: take 2

Decide whether an array A of natural numbers contains i.

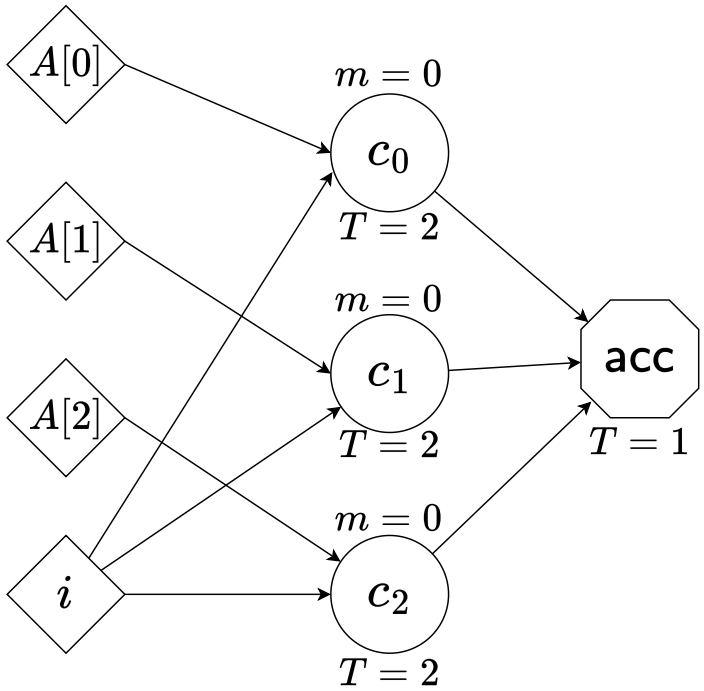


Concept: i is now offered as input using a single spike with delay i; A and the computation are encoded in the machine

Energy: $\leq |n| + 3$ spikes needed

Solving a simple problem: take 3

Decide whether an array A of natural numbers contains i.



Concept: both the search value and the integers in the array are offered as input to the machine, while the size of the array is fixed

Energy: $\leq 2|n| + 2$ spikes needed

Generality and resources of SNN computations

- Observation 1: we can do *program* an SNN to do basic computations such as finding an integer in an array

Generality and resources of SNN computations

- Observation 1: we can do *program* an SNN to do basic computations such as finding an integer in an array
- Observation 2: computation and data can be co-located in the network and become morphed

Generality and resources of SNN computations

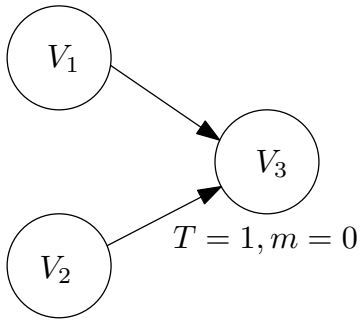
- Observation 1: we can do *program* an SNN to do basic computations such as finding an integer in an array
- Observation 2: computation and data can be co-located in the network and become morphed
- Observation 3: timing (when does a spike arrive) using delays is crucial in the computation

Generality and resources of SNN computations

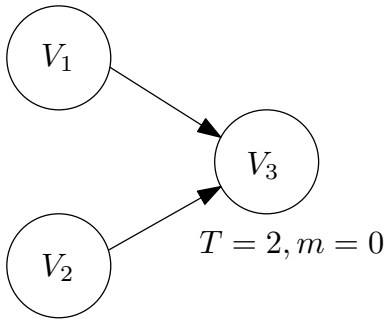
- Observation 1: we can do *program* an SNN to do basic computations such as finding an integer in an array
- Observation 2: computation and data can be co-located in the network and become morphed
- Observation 3: timing (when does a spike arrive) using delays is crucial in the computation
- Observation 4: there is a trade-off between generality of the network and resources needed for the computation
- Observation 5: the final example looks quite like a Boolean circuit family, with different circuits for each input size - can we do more generic computations (rather than non-uniform)?

A family of SNNs simulates a Boolean circuit family

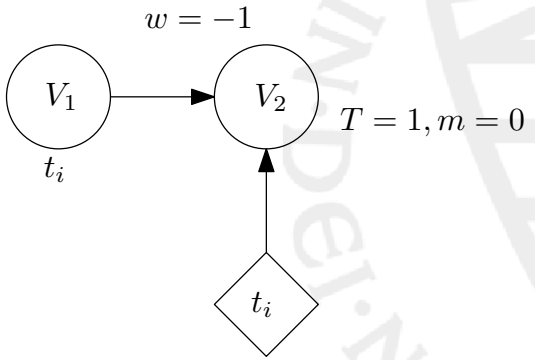
- We can simulate binary operations with simple circuit components when we consider a spike at a particular moment to indicate a 1 and the absence of a spike as a 0:



$V_3 = V_1 \vee V_2$



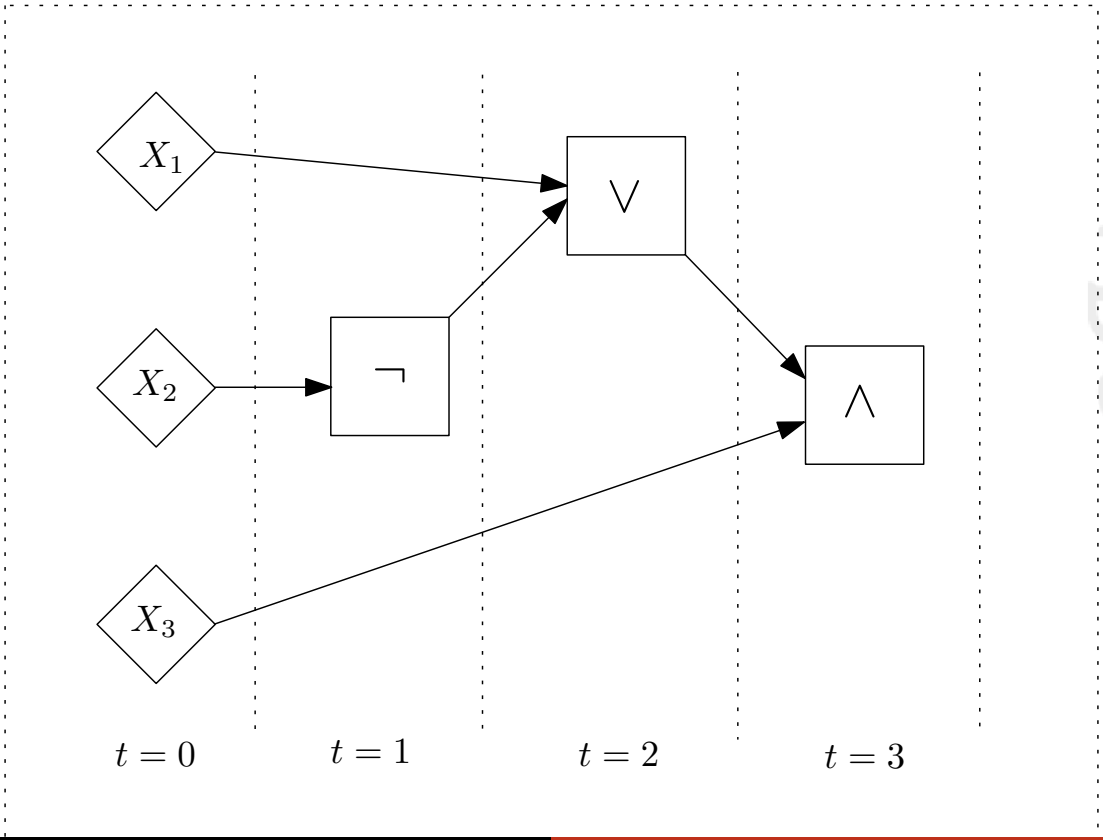
$V_3 = V_1 \wedge V_2$



$V_2 = \neg V_1$

A family of SNNs simulates a Boolean circuit family

- Simulating $(x_1 \vee \neg x_2) \wedge x_3$ with input spikes (or absence of it) at $t = 0$ using these components. Spikes need to be delayed to arrive at the appropriate time step at the 'gate'.



Turing complete?

- A non-uniform family of SNNs can compute Boolean circuits

Turing complete?

- A non-uniform family of SNNs can compute Boolean circuits
- Can a *single* SNN, given an input string, do any computation?
- In order words: are SNNs Turing-complete?

Turing complete?

- A non-uniform family of SNNs can compute Boolean circuits
- Can a *single* SNN, given an input string, do any computation?
- In order words: are SNNs Turing-complete?
- Maass (1996): Yes, assuming **unlimited** precision of weights and potentials

Turing complete?

- A non-uniform family of SNNs can compute Boolean circuits
- Can a *single* SNN, given an input string, do any computation?
- In order words: are SNNs Turing-complete?
- Maass (1996): Yes, assuming **unlimited** precision of weights and potentials
- Cabessa and Siegelmann (2014): Yes, with synaptic plasticity

Turing complete?

- A non-uniform family of SNNs can compute Boolean circuits
- Can a *single* SNN, given an input string, do any computation?
- In order words: are SNNs Turing-complete?
- Maass (1996): Yes, assuming **unlimited** precision of weights and potentials
- Cabessa and Siegelmann (2014): Yes, with synaptic plasticity
- Likely, for bounded precision and no plasticity, an SNN has the same complexity as a finite state automaton

Setting up a complexity framework

- We want to study both structural and computational complexity questions relative to SNNs

Setting up a complexity framework

- We want to study both structural and computational complexity questions relative to SNNs
- e.g. structural: is there something like an energy hierarchy?
- computational: are there problems that are complete for some complexity class?

Setting up a complexity framework

- We want to study both structural and computational complexity questions relative to SNNs
- e.g. structural: is there something like an energy hierarchy?
- computational: are there problems that are complete for some complexity class?
- We need to build a framework: decide on a machine model, identify resources and hierarchies of complexity classes

Setting up a complexity framework

- We want to study both structural and computational complexity questions relative to SNNs
- e.g. structural: is there something like an energy hierarchy?
- computational: are there problems that are complete for some complexity class?
- We need to build a framework: decide on a machine model, identify resources and hierarchies of complexity classes
- We will start with studying some formal properties of resources and defining different machine models!

Resources

- Turing machine (TM): space (number of cells on the tape used) and time (number of state transitions until completion)

Resources

- Turing machine (TM): space (number of cells on the tape used) and time (number of state transitions until completion)
- SNN: space (number of neurons in the model), time (number of time steps until acceptance), and **energy** (number of spikes until completion)
- Completion: we assume the presence of a dedicated accept and reject neuron that halt the computation

Resources

- Turing machine (TM): space (number of cells on the tape used) and time (number of state transitions until completion)
- SNN: space (number of neurons in the model), time (number of time steps until acceptance), and **energy** (number of spikes until completion)
- Completion: we assume the presence of a dedicated accept and reject neuron that halt the computation
- Note these resource definitions are all abstractions from the physical world (e.g., no spike $\neq$ no energy)

Resources

- Turing machine (TM): space (number of cells on the tape used) and time (number of state transitions until completion)
- SNN: space (number of neurons in the model), time (number of time steps until acceptance), and **energy** (number of spikes until completion)
- Completion: we assume the presence of a dedicated accept and reject neuron that halt the computation
- Note these resource definitions are all abstractions from the physical world (e.g., no spike $\neq$ no energy)
- We can prove some general properties about these resources

Resources

- Energy is upper bounded by time $\times$ space: a neuron can spike at most once in each time step

Resources

- Energy is upper bounded by time $\times$ space: a neuron can spike at most once in each time step
- Note: space resource is, unlike TMs, by definition independent of the input; an SNN does not have a 'working tape' and in a TM we do not consider the number of states as a resource.

Resources

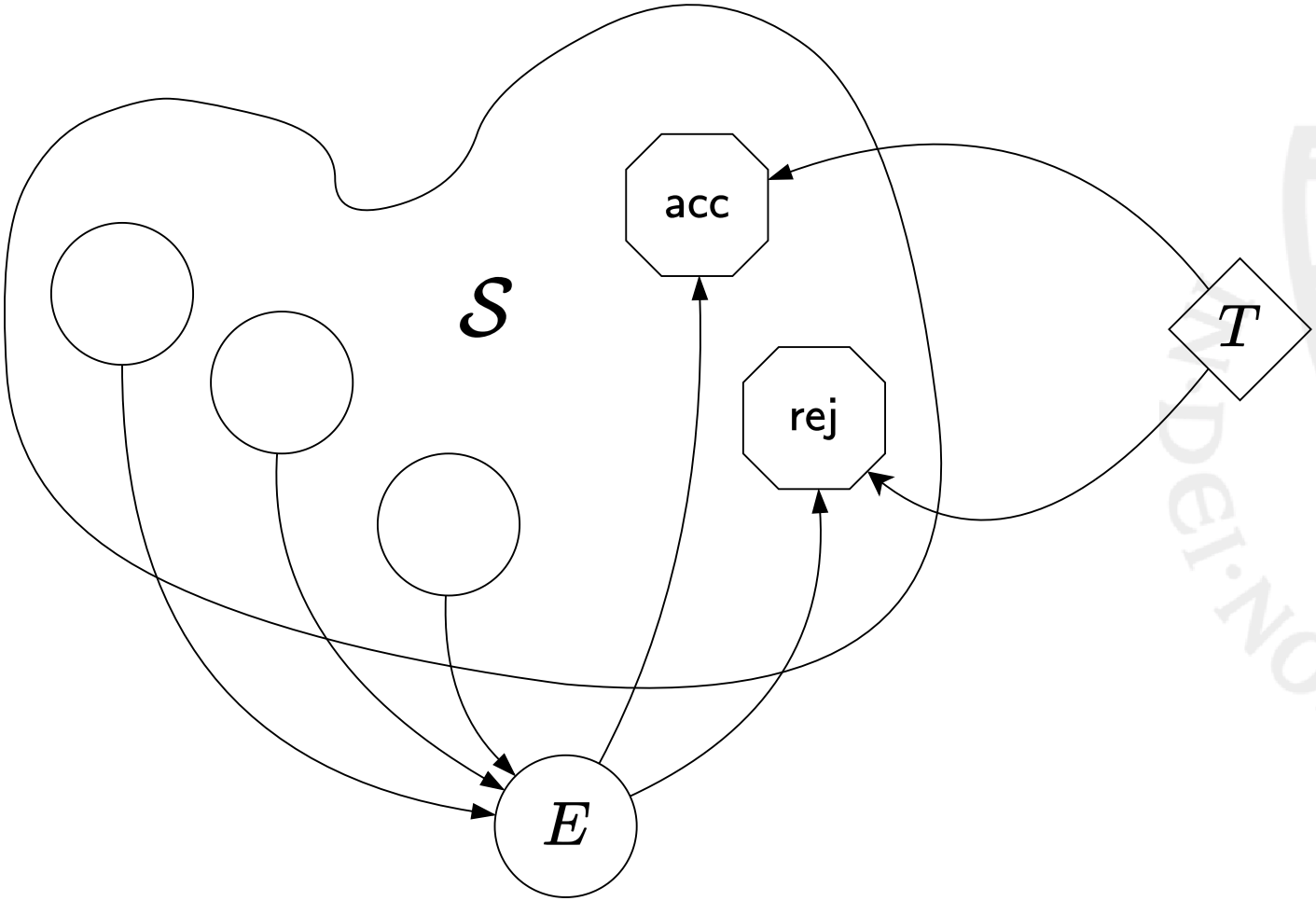
- Energy is upper bounded by time $\times$ space: a neuron can spike at most once in each time step
- Note: space resource is, unlike TMs, by definition independent of the input; an SNN does not have a 'working tape' and in a TM we do not consider the number of states as a resource.
- We use a **spike train** as read-only, uncontrolled input stream
- We use dedicated accept and reject neurons; a computation starts at $t = 0$ at the first instance of the spike train and ends when either neuron fires

Resources

- Energy is upper bounded by time $\times$ space: a neuron can spike at most once in each time step
- Note: space resource is, unlike TMs, by definition independent of the input; an SNN does not have a 'working tape' and in a TM we do not consider the number of states as a resource.
- We use a **spike train** as read-only, uncontrolled input stream
- We use dedicated accept and reject neurons; a computation starts at $t = 0$ at the first instance of the spike train and ends when either neuron fires
- We can limit computation time and energy usage (and go in reject mode if exceeded) using a **clock** and a **meter**

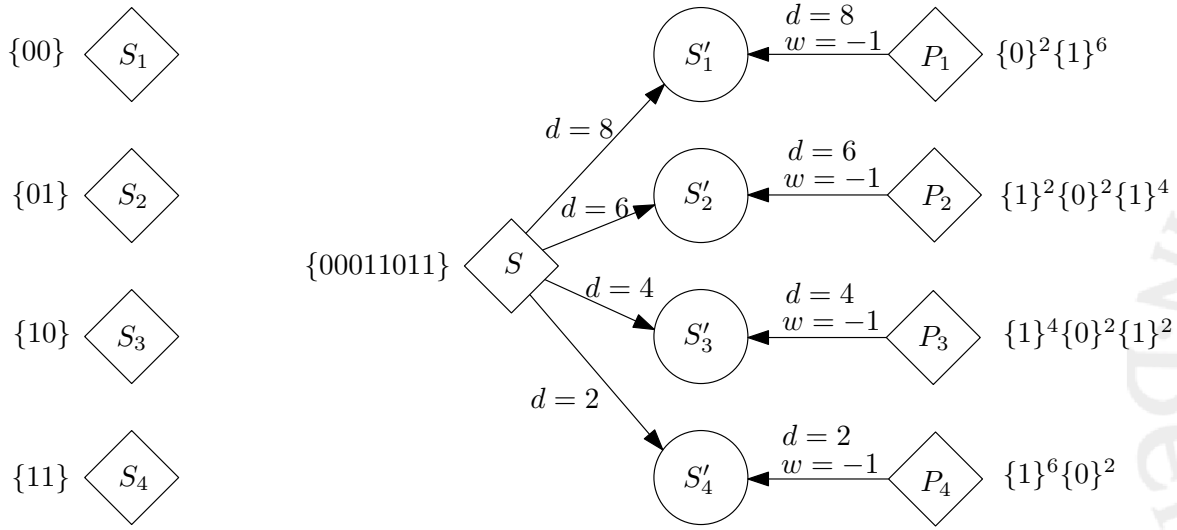
Clock and meter

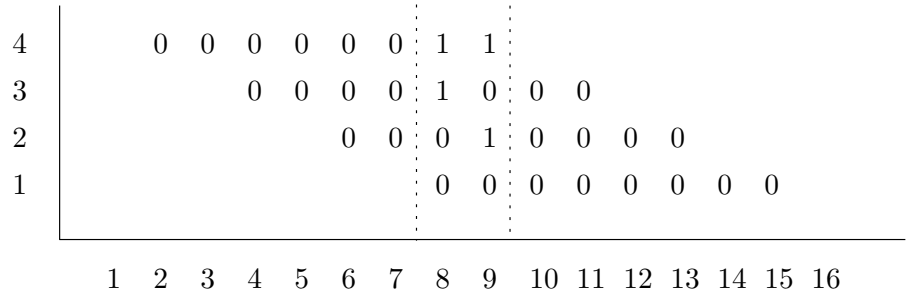
- Adding a clock and a meter to constrain computations



Generalisation: spike wave

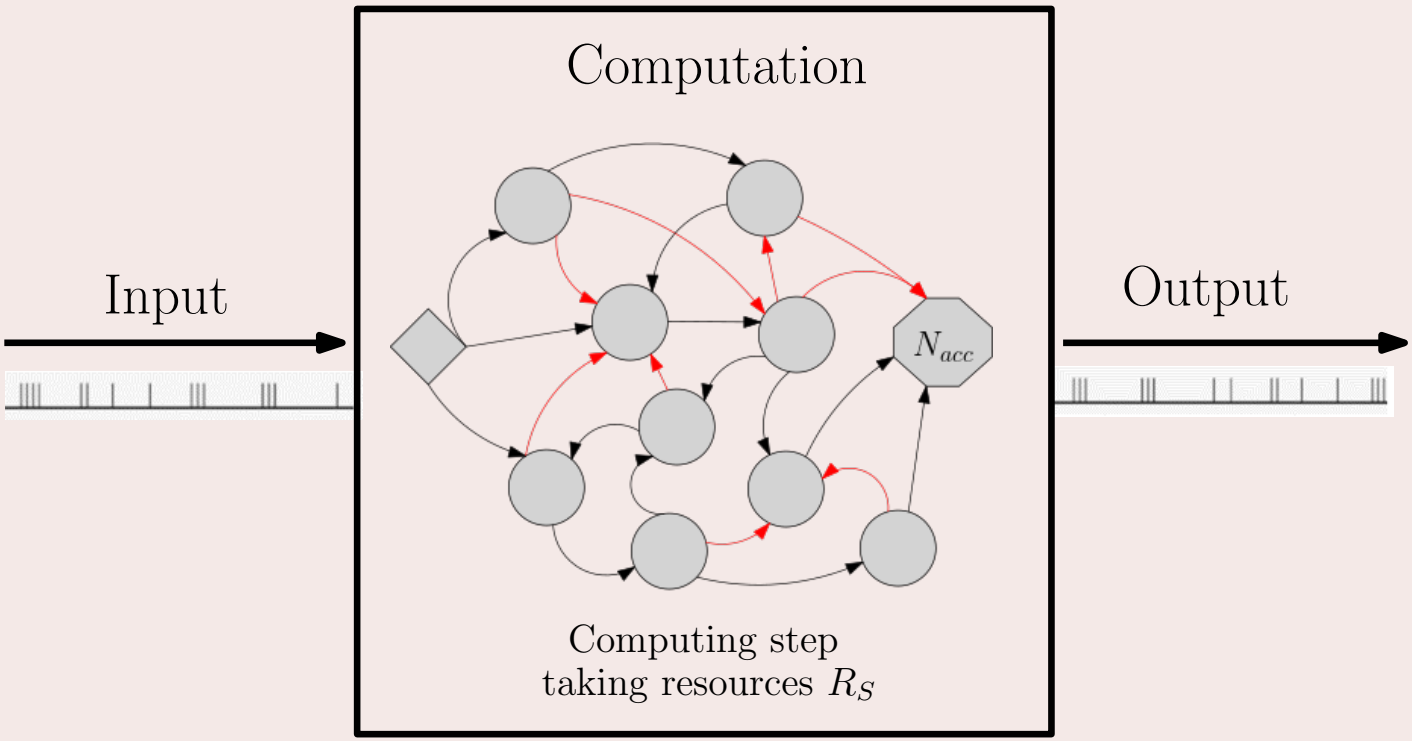
- Some generalisation: a spike wave, rather than a single input, can be simulated by a single stream





Machine models (1)

Spike-input model



Spike-input model

- This model assumes a given SNN with varying input stream

Spike-input model

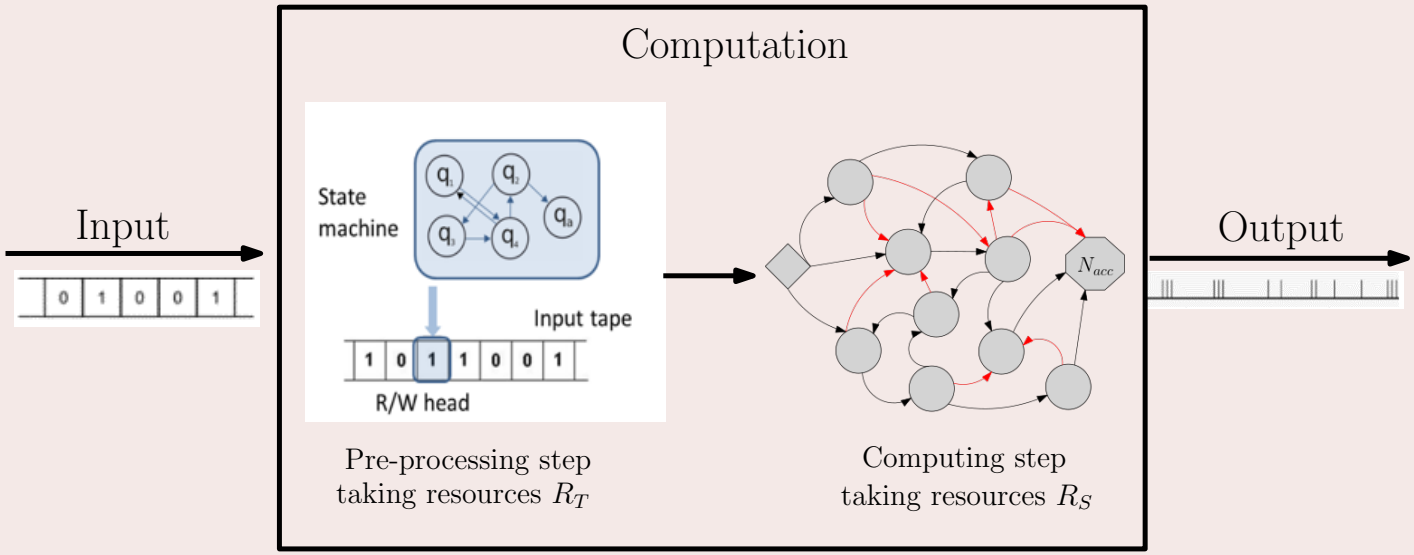
- This model assumes a given SNN with varying input stream
- It could compute, for example, whether an array of a given (fixed) size contains an number, where the numbers are given as input and accepts if it is, and rejects if it is not
- This is quite similar to a Boolean circuit

Spike-input model

- This model assumes a given SNN with varying input stream
- It could compute, for example, whether an array of a given (fixed) size contains an number, where the numbers are given as input and accepts if it is, and rejects if it is not
- This is quite similar to a Boolean circuit
- A *family* of SNNs can be powerful, but as a singleton machine it is (given finite precision) effectively a finite state automaton
- It also does not really combine data and algorithm in one model and as such, does not harnass the power of SNNs

Machine models (2)

Pre-processing model



Pre-processing model

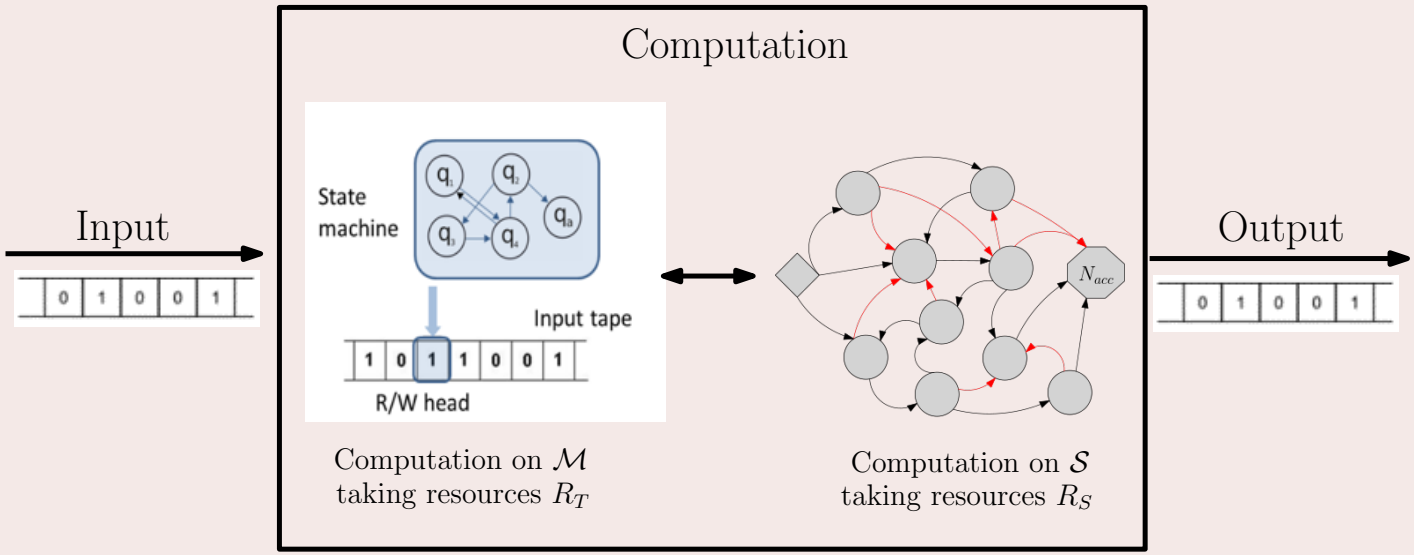
- This models a traditional algorithm, on input, constructing an SNN and delegating the computation to this machine

Pre-processing model

- This models a traditional algorithm, on input, constructing an SNN and delegating the computation to this machine
- It can be considered to be an extension of the advice function in non-uniform complexity theory where we map $I = \{1\}^n$ to a circuit that can deal with inputs of size n
- Here, we relax this constraint and can map any input $I = \{0, 1\}^n$ to a spiking neural network
- With polynomial resources for time and space for the mapping as well as for the SNN, this can be shown to be equal to P

Machine models (3)

Interactive model



Interactive model

- This model is effectively a neuromorphic co-processor idea where we can, at any time, construct a network and offload some computation

Interactive model

- This model is effectively a neuromorphic co-processor idea where we can, at any time, construct a network and offload some computation
- For example, computing shortest paths as sub-routine of computing a maximum network flow (MSc thesis project Abdullah Ali, 2019)
- Interesting idea, but..
  - We don't actually have such co-processors
  - We need to construct each network from scratch (the ability to modify an existing network has not been considered yet)
  - In practice this will be a massive overhead and really different from an FPU / GPU operating in the same memory space

No magic in neuromorphic computing..

- Conjecture: for fixed precision, spike-input machine can be simulated by a finite state automaton as we have no memory or stack

No magic in neuromorphic computing..

- Conjecture: for fixed precision, spike-input machine can be simulated by a finite state automaton as we have no memory or stack
- Proven in the paper: for polynomially constrained resources for the SNN, both the interactive and the pre-processing classes equal P

No magic in neuromorphic computing..

- Conjecture: for fixed precision, spike-input machine can be simulated by a finite state automaton as we have no memory or stack
- Proven in the paper: for polynomially constrained resources for the SNN, both the interactive and the pre-processing classes equal P
- More interesting: look into constrained resources for either the TM or the SNN
- For example, we can show that the P-complete Network Flow problem can be solved in Logspace with access to an SNN with linear resources

..but for some instances it may help!

- Brad Aimone and colleagues (2020) showed that finding a shortest path can be more efficient on a neuromorphic device than on a regular TM given constraints on a) the edge weights (logarithmic in number of nodes) and b) the number of edges in the shortest path (square root of number of edges)

..but for some instances it may help!

- Brad Aimone and colleagues (2020) showed that finding a shortest path can be more efficient on a neuromorphic device than on a regular TM given constraints on a) the edge weights (logarithmic in number of nodes) and b) the number of edges in the shortest path (square root of number of edges)
- Basically code edge weights as delays, inhibit neuron after the first incoming spike, and keep track of the path
- For some problem instances, the inherent parallelism and time-dependent nature of the network works very well!

..but for some instances it may help!

- Brad Aimone and colleagues (2020) showed that finding a shortest path can be more efficient on a neuromorphic device than on a regular TM given constraints on a) the edge weights (logarithmic in number of nodes) and b) the number of edges in the shortest path (square root of number of edges)
- Basically code edge weights as delays, inhibit neuron after the first incoming spike, and keep track of the path
- For some problem instances, the inherent parallelism and time-dependent nature of the network works very well!
- But, you can construct adversarial instances where the delays become the bottleneck

Completeness results

- So far, only completeness for halting questions ("given an input, does the network accept the input within given resource bounds")

Completeness results

- So far, only completeness for halting questions ("given an input, does the network accept the input within given resource bounds")
- Completeness for time does not really make sense - but most interesting part would be completeness for energy!
- Can we find an inherently energy-hungry problem?

Completeness results

- So far, only completeness for halting questions ("given an input, does the network accept the input within given resource bounds")
- Completeness for time does not really make sense - but most interesting part would be completeness for energy!
- Can we find an inherently energy-hungry problem?
- .. and could we find differences in energy needs in SNN and TM models?
- Conjecture: existence of energy hierarchy (with more spikes, strictly more can be computed)

Conclusion

- What do we have: different machine models and different complexity classes / hierarchies based on them
- Some basic first results and constructs
- No real breakthroughs, unfortunately..
- .. due to lack of resources (grants) and small community
- What will be the future of neuromorphic computing? So far, no killer application, little commercial interest, does not yet *really* live up to expectations!