Cellular Automaton

And

Spiking Cellular Automaton

Andrew D Newman

University of York

February 2021

3rd Year Project Initial Report for degree of BEng in Electronic Engineering

# Introduction

Neurons in the brain communicate with one another by sending spikes of voltage along synapses that connect them to other neurons. The artificial neural nets typically in use at the time of writing [1] use a single value to represent the density (per unit time) of spikes being transmitted by each neuron. Biologically inspired spiking neural networks have been shown to also encode information in the timing of the spikes [1] [2], and in the delay between a spike being fired from one neuron and received by another [3] [4].

A cellular automaton is a regular grid of discrete cells in which the state of each cell depends on the state of its neighbours, so that the cells change state over time. A popular cellular automaton is Conway’s Game of Life [5] in which each cell is either ‘alive’ or ‘dead’ and changes between those states depending on the number of live immediate neighbours it has. Complex behaviour emerges from the simple rules.

The goal of this project is to make a cellular automaton in a similar style to Conway’s Game of Life but using a biologically inspired spiking neuron model for the cells. It is hoped that this will lead to interesting behaviour of the automaton.

This report outlines our intended method for investigating the idea of a Spiking Neural Network (SNN) Cellular Automata (CA), and reviews information about SNNs from a variety of sources.

# Objectives

This project will be entirely software oriented. Where possible platform independence will be aimed for, but the Microsoft Windows OS on an x64 architecture will be the primary target.

The software will display an automaton in a regular grid of cells in which the behaviour of each cell is determined by the state of its neighbours. The general goal of an automaton of this nature is to demonstrate complex behaviour emerging from a simple set of rules.

Conway’s objectives for the Game of Life [5] will serve as a starting point for our objectives for our own automaton:

1. “There should be no initial pattern for which there is a simple proof that the population can grow without limit.”
2. “There should be initial patterns that apparently do grow without limit.”
3. “There should be simple initial patterns that grow and change for a considerable period of time before coming to end in three possible ways: fading away completely (from overcrowding or becoming too sparse), settling into a stable configuration that remains unchanged thereafter, or entering an oscillating phase in which they repeat an endless cycle of two or more periods.”

To these we will add our own objectives:

1. Information flow between cells must be in the form of spikes similar in nature to those conveyed by axons in biological nervous systems.
2. The behaviour of the automaton should be distinguishable from behaviours which are possible with a 1st or 2nd generation neural net.
3. There will be a deliverable from the project in the form of a piece of software that can be used to experiment with different neuron models and configurations in an automaton.
4. It will be possible to easily configure the software to run an automaton which meets objectives 1 to 5.

We consider objective 1 to be rather vague (especially since the game of life itself has some simple examples that do grow without limit [5]) but its intention will serve as a guideline.

We have no proof that objective 5 is possible but will aim for it.

# Ethics

We do not anticipate any ethical issues arising in this project, beyond those typical of any other software project. This specific research will make only sporadic use of one computer.

From an environmental point of view it may be worth noting that neural nets are very processor intensive and that if projects of this nature became common place we would expect them to drive a similar energy consumption pattern to that seen in bit coin mining (which is more energy intensive than actual mining [6]). Purpose built hardware should mitigate this effect to some extent.

# Overview

## Biological Neurons

A neuron is a living cell that has many points of input (called dendrites) and a single output (called an axon) [1]. The neuron is affected by a number of ion channels that allow the concentrations of Sodium (NA), Potassium (K), Calcium (Ca) and Chloride (Cl) ions to change over time, at rates which depend not only on their current concentrations, but also influenced by input voltages applied through the dendrites and the presence of chemicals such as neurotransmitters. A neuron responds to changes in the concentrations (often but not always a potential threshold being crossed) by generating spikes of voltage on their axon. The axons connect to the dendrites of other neurons via synapses. Cells in the brain typically have around 10,000 synaptic connections, up to as many as 100,000 in some cases.

A picture containing text, vector graphics

Description automatically generated

Figure Structure of a Typical Neuron

dhp1080 - Creative Commons License - <https://commons.wikimedia.org/wiki/File:Neuron.svg> – accessed 2021-01

In the 1950s a model of the behaviour of a giant squid neuron was developed through experiment [2] [7]. Known as the Hodgkin Huxley Model it is a 4-dimensional family of exponential differential equations, describing the changes in state of the neuron and its potentials over time. These equations have been used to describe other neurons in other species and a wide variety of configurations exists [2, pp. 46-48] leading to different behaviours, both in how a neuron becomes excited enough to trigger a spike, and the shape and pattern of the spikes generated.

The giant squid neuron is also not representative of all the possibilities. “At least a dozen types of ion channel can be involved in the spike generation of human neocortical neurons.” [1, p. 33].

Examples of distinctly different spiking behaviours are neurons which produce a single spike, or a sequence of repeating spikes, or a rapid burst of spikes followed by a quiet period. While many neurons fire in response to a potential threshold being crossed in response to input spikes it is by no means all [2, pp. 238-239], some respond to incoming spikes only at the correct resonant frequency [2, pp. 232-235] and there are others which emit spikes in response to a *decrease* in their inputs [2, pp. 244-246].

Information flows from one neuron to another in the number of spikes transmitted, the time between the spikes [1] [2], and the time taken for them to travel down the axon [3] [4]. To what extent each of these is significant is still open to debate, but all of them have been shown to increase the theoretical computational power of a neuron.

## Spikes

The response of a neuron to an excitatory input is often modelled as a sigmoid curve, which is useful when doing mathematical analysis, but not representative of what is seen, since it doesn’t inherently show the output rapidly falling back to zero. The spikes generated by neurons tend to show a rapid rise in voltage followed by a slower exponential decay [8] but there is quite a bit of variety possible even within a single neuron, depending on the input it receives.

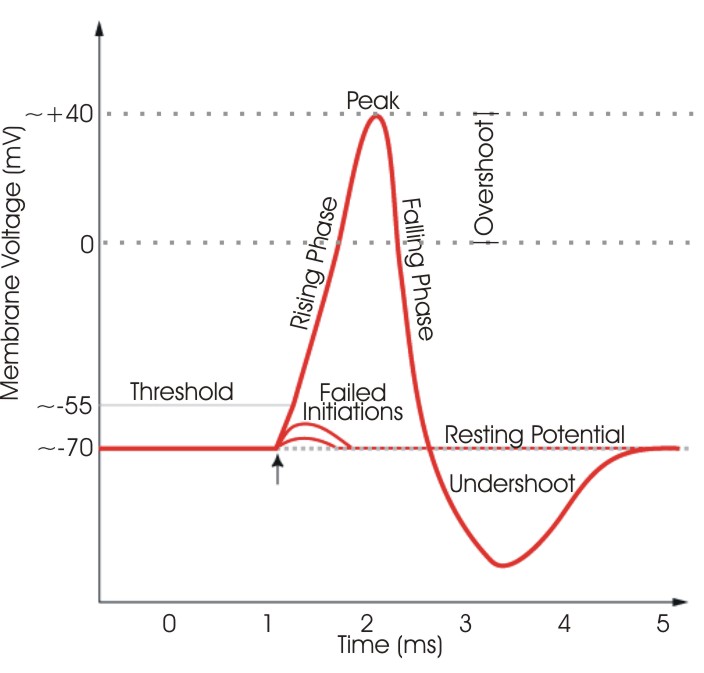


Figure Action Potential

Original author Synaptidude at English Wikipedia. GFDL 1.2, <https://commons.wikimedia.org/w/index.php?curid=6030880>. Accessed 2021-01-31

Typical spikes have a peak voltage of the order of a few millivolts and last for up to a few tens of milliseconds. The neuron producing the spike will generally have a refractory period following it during which it will be unable, or at least very reluctant, to produce another spike.

## Phase Diagrams and Bifurcations

If the state of a neuron, or indeed anything, can be represented with N variables, then it can be considered as a point in an N dimensional space of possible states. If the behaviour can be represented as a rate of change over time of the N variables then the behaviour can be considered as a set of N dimensional vectors within that space, each indicating the direction and magnitude of the differentials at a point [2, pp. 8-20]. A diagram of these arrows is called a Phase Space plot.

If all the arrows on the plot near a point tend towards it then that is a stable point of equilibrium. Small perturbations to the state will not result in lasting changes. If the arrows all tend away from a point it is an unstable equilibrium. If the arrows tend towards a point along one arbitrary axis but away from it along another it is called a saddle. A set of arrows forming a loop is called a limit cycle attractor.

We will consider two-dimensional phase spaces here, for simplicities sake, but remain aware that biological neurons have more (Hodgkin Huxley has 4 for example). A diagram a 2D phase space is shown in Figure 3, showing a saddle-node bifurcation. The system will rest at the stable equilibrium, and small inputs will have no lasting effect, but any change to the state moving it to the right of the saddle node will cause a significant change in the behaviour, possibly resulting in the limit cycle being reached.

If this diagram were to represent a neuron with the potential on the horizontal axis then we can see we have a coincidence detector that ignores small increases in potential but, on receiving one large enough to cross an internal threshold, begins firing spikes repeatedly until inhibited by an external influence.

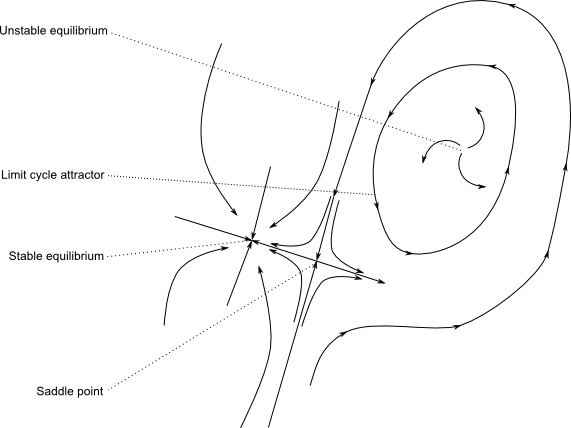


Figure Features of a 2D Dynamical System (recreated in the style of Izhikevich [2, pp 17])

There are four possible kinds of bifurcation in such a system [2, p. 11], depending on whether there exists a saddle node, and whether there exists a limit cycle. For a single model of neuron behaviour to be able to exhibit a good range of different types of behaviour it needs to be configurable to act as any of the four. Those without limit cycles fire only one spike in response to excitation, and those with no saddle node act as resonators instead of integrators.

## Computational SNNs

The Hodgkin Huxley model is complicated and correspondingly slow to simulate. Faster approaches to it have been proposed [9] and it is interesting from a biological point of view, but for a large and/or fast neural network in software we need to look for alternatives.

There have been many SNN models developed for this purpose, with different goals and advantages [10] [11] [12].

A simple and well-known model is Leaky Integrate and Fire (LIF) of which there are several variants. Each neuron has a potential that decays (leaks) over time, incoming spikes increase it, and a spike is generated if the potential passes a predefined threshold. An LIF neuron with no leakage is an integrator – it has a potential which is the sum of all incoming spikes over time. An LIF neuron which leaks extremely fast is a coincidence detector – it has a potential which will only cross the threshold if multiple spikes are received simultaneously [1] [2]. An integrator is essentially the same principle as a 2nd generation neuron (it cares only about the number of spikes emitted, not the timing of them) and as such probably cannot be used to meet our 5th objective (behaviour distinct from that of a 2nd generation network). LIF neurons also cannot reproduce many other behaviours of their biological counterparts, such as resonance and bursting – they cannot form the four bifurcation types.

An equation for a simple LIF neuron, excluding tests for the threshold being exceeded, is:

An often referenced alternative is a 2D model named after its creator Izhikevich [13] which is able to exhibit 20 different spiking behaviours using two simultaneous differential equations, one quadratic and one linear, and can be an integrator, coincidence detector or resonator, which makes it far more interesting for our automaton. Izhikevich can behave as any of the four bifurcation types. Furthermore, we can see no way to create such a system using less than 2 variables, and we doubt it is possible with two variables without at least one quadratic (because a bifurcating system needs two nodes and we don't get that without two solutions to our equation) which means the Izhikevich model may be the simplest possible to meet our requirements.

The dynamic state of an Izhikevich neuron (again excluding threshold responses) is:

Kumar etal [12, pp. 24-30] present an alternative model to Izhikevich, again with only two equations. They refer to Izhikevich as inspiration and provide a similar set of possible output forms. It is not clear why this model might be preferred over Izhikevich, but it demonstrates biologically similar behaviours in response to short pulses of DC current, where Izhikevich seems inclined to use a unit step function as his input. This is encouraging, since we will be using spikes as our inputs and a DC pulse is more representative. The equations used include an exponential, which makes them a little more complicated, but if they also give better results it will be worth it.

## Hardware SNNs

IBM have developed a True North architecture [14] with the goal of creating a neuron model that can be effectively implemented in custom hardware "sufficient to support useful and interesting cognitive algorithms, while the cost should be no more than necessary in terms of power, area, and speed".

They state "we were able to qualitatively replicate the 20 behaviors of the Izhikevich dynamical neuron model using a small number of elementary neurons" which is interesting. It implies a simpler model that has the Izhikevich model emergent from it, and emergent behaviour is a good thing in cellular automatons.

The algorithm works entirely in integer (fixed point) arithmetic, which is great for hardware implementations but means we may need a different kind of synapse matrix and spike to implement it in our own code, which could prove awkward. Fixed point maths is possible in software but not generally built into the available languages. We might assume that using floats instead will be close enough.

## Challenges Presented by Cellular Automata

Our automata will not learn, which excludes a large amount of the complexity of neural nets from the project. However, other issues do need to be addressed.

For the automaton to be "interesting" it needs to avoid quickly collapsing into a quiescent state (see objective 3). For this to be true the number of neurons which fire as a result of N other neurons firing must average at least N. However, if the average is greater than N then we can expect the automaton to quickly saturate instead, with everything firing as fast as it is able, which is also uninteresting (and breaks objective 1). Life avoids this by killing any cell that has 4 or more neighbours, but it does not seem appropriate to directly prevent a neuron from firing when it is overstimulated. Instead we expect to need inhibitory behaviours to maintain balance. These can be simple negatively weighted inputs (subtractive), “shunting” in which a connection decreases the effectiveness of excitatory inputs (divisive) [1, p. 22], or fatigue [1, p. 35].

The potential of a neuron in most of the described models can grow without limit and is only prevented from doing so by the reset that follows a spike being fired. However, although the Izhikevich model can act as an inhibitory neuron and even mimic the output of the biological ones in the brain, there does not appear to be anything limiting the negative potential a neuron can have. This will not be a problem with shunting inhibitions, but otherwise seems to part of a common theme through all the literature reviewed here - the existence of inhibitors, whether neurons or otherwise, is acknowledged but mostly ignored. We suspect this is because most work on neural nets has been geared towards forward connected nets that respond to an input event, rather than a continuously updating system such as our automaton that needs to self-regulate.

IBM make many arguments for combining multiple kinds of neuron [14] and our design will account for this from the start, and should be able to incorporate the need for inhibitory neurons.

There are many references to stochastic integration (a random chance of synapses being integrated) and noise, with noise being given as an explanation for the speed of the brains response to visual stimuli [1] [2][14]. Unfortunately, any random influence will break repeatability in our automaton. The same pattern of inputs to an automaton would usually result in the same behaviour, regardless of where or when it occurs, and we would rather respect that than take the biologically more realistic stochastic approach. Repeatability will also make debugging it easier. It will not be difficult to add a random factor at a late stage if it proves desirable.

## Performance

An effective automaton needs to be fairly large, in order to provide enough space for interesting behaviours to emerge. This means it also needs to be fairly fast, so as not to take an unacceptable amount of time to execute. No precise definition of “fairly large” or “fairly fast” is available.

Izhikevich claims a 1ms simulation time for 10,000 neurons with 1,000,000 synapses on a 1GHz PC [13]. That is only a 100x100 grid of neurons, which is rather small for an automaton and would limit the size of feature we could hope to detect. It is also only 100 synapses per neuron, rather than the 10,000 typical of the brain [1]. There is a balance to be made between designing for speed and premature optimization and we will try to be reasonable, but it is obvious even without profiling that efficiency will be an important factor here [10] [11].

Note that an automaton of this nature is extremely parallelizable and any attempt to fully optimize it should almost certainly be targeting a GPGPU. This project needs to be flexible and will therefore target the CPU, and consequently the investigative phase will only aim to be fast enough to be usable, rather than as fast as possible.

# Requirements Specification

Because we begin this project with no idea what kind of SNN will allow us to meet the objectives, or how it will need to be configured or connected, the software we produce will need the flexibility to run many different configurations to allow us to search for one.

1. The software implementation will be organized according to the Model/View/Controller pattern
2. The software will allow different sizes of automaton to be modelled
3. The software will allow the simulation to be run at different speeds, stopped, or stepped one frame at a time
4. The software will allow selection of different neuron models
   1. There will be a LIF (Leaky Integrate and Fire) model
   2. There will be an Izhikevich model [13]
   3. There may be other models
5. The software will allow easy configuration of the selected model
   1. Any appropriate parameters will be made available for change via the GUI
   2. It will be possible to persist interesting combinations of parameters for reuse
6. The software will allow different spike shapes to be configured
7. The software will allow different connection patterns of neurons via synapses to be configured
   1. There will either be a synapse matrix editor built into the software or it will have the ability to load synapse matrices from image files
8. The software will display an animation of the automaton as it progresses
   1. The software may provide alternative display styles such as spikes vs neuron potentials
9. The software will allow different starting states to be configured.
   1. There will either be a state editor built into the software or it will have the ability to load states from image files
10. The software will allow a given starting state to be reused
    1. A given starting state, used multiple times with the same model and configuration, will give the same progression and end state each time

## Design Sketch

For reasons of speed and familiarity the controller will be written in C++. For simplicities sake the View will therefore also use C++. We expect to use Qt for the GUI in the View but will refrain from making the controller dependent on it.

Although we believe inhibitory neurons will be needed, and mixing multiple neuron types is likely to be a good idea [14], we feel quite strongly that an automaton should be a uniform grid of cells if possible, and it is not clear what pattern of different neuron types would be appropriate to insert. Instead we will allow the creation of multiple layers of neurons. A layer of inhibitory neurons can then be included if desired, for example.

For defining the synaptic connections between neurons, we will use another 2-dimensional array of synapses, each specifying a weight and time delay for its associated connection (see Figure 4). We will refer to this as a Synapse Matrix. It has similarities to the Convolution Matrix commonly used in graphics editing packages [15].

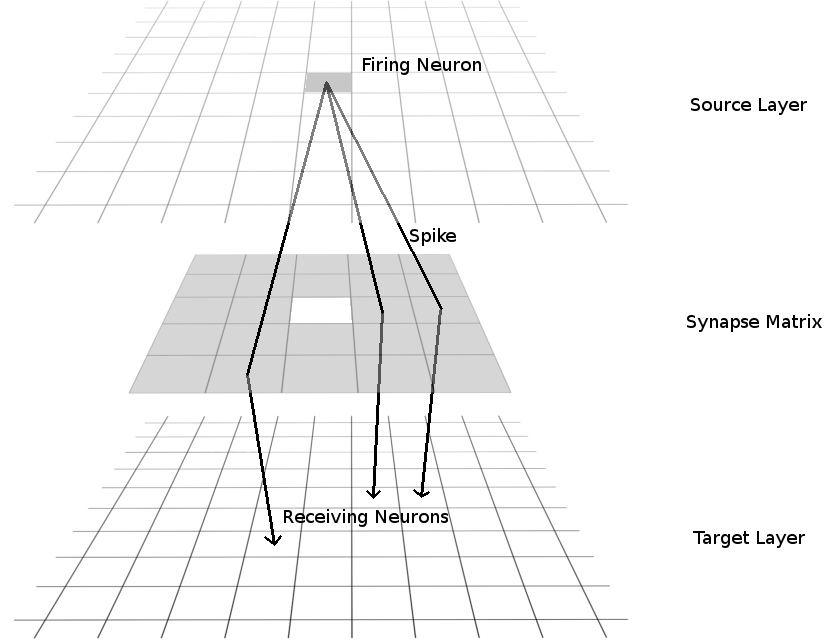


Figure A synapse matrix connecting two layers of neurons

It will be possible for the source and target layer to be the same.

In the interests of efficiency, we will fire spikes along synapses using an event-based mechanism instead of polling. This has been used to decrease power consumption in True North [14] and will serve to decrease CPU usage for us.

We will be displaying the state of the network as it progresses, unless that proves to be unachievable. Whether we display a representation of the internal state of the neurons, or display the spikes that fire, or both, remains open pending experiment.

# Timetable

|  |  |  |
| --- | --- | --- |
| Phase | Description | Target Delivery Date |
| Research | Reading and reviewing literature related to SNNs and writing this document. | 2021-02-04 |
| Interface | Building a GUI to use for experimenting with different models and parameters. | 2021-02-25 |
| Modelling | Building the models and experimenting with them to find an automaton that meets the objectives. | 2021-03-25 |
| Finalization | Decide on and precisely specify an automaton ruleset that gives good results. Potentially reimplement in a specialized form, depending on time and characteristics. | 2021-04-20 |
| Demonstration | Prepare for and give a presentation on the project. | 2021-05-04 |
| Report | Finish and deliver project report. | 2021-05-13 |

This project will take an agile approach, on account of not knowing where it will end up. I am confident of the timetable above regarding the interface phase, but modelling is more open ended, and finalization could take anything from 1 day to 30 depending on what is needed when we get there.

# References

|  |  |
| --- | --- |
| [1] | T. P. Trappenberg, Fundamentals of Computational Neuroscience, Truro: Oxford University Press, 2002. |
| [2] | E. M. Izhikevich, Dynamical Systems in Neuroscience, The MIT Press, 2007. |
| [3] | W.Mass, “Networks of Spiking Neurons: The Third Generation of Neural Network Models,” Elsevier Science Ltd., Graz, 1996. |
| [4] | A. Taherkhani and etal, “A review of learning in biologically plausible spiking neural networks,” *Neural Networks,* vol. 122, pp. 253-272, 2020. |
| [5] | M. Gardner, “MATHEMATICAL GAMESThe fantastic combinations of John Conway's new solitaire game "life",” *Scientific American,* pp. 120-123, 10 1970. |
| [6] | M. J. Krause and T. Tolaymat, “nature.com,” 05 11 2018. [Online]. Available: https://www.nature.com/articles/s41893-018-0152-7. [Accessed 12 2020]. |
| [7] | A. L. Hdgkin and A. F. Huxley, “A QUANTITATIVE DESCRIPTION OF MEMBRANE CURRENT AND ITS APPLICATION TO CONDUCTION AND EXCITATION IN NERVE,” *Journal of Physiology,* vol. 117, pp. 500-544, 1952. |
| [8] | G. Eyal and etal, “frontiers in cellular neuroscience,” 29 June 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncel.2018.00181/full. [Accessed 01 2021]. |
| [9] | A. M. Nagya and N. H. Sweilamb, “An efficient method for solving fractional Hodgkin–Huxley model,” *Physics Letters,* vol. 378, pp. 1980-1984, 2014. |
| [10] | Eugene.M.Izhikevich, “Which Model to Use for Cortical Spiking Neurons?,” *IEEE Transactions on Neural Networks,* vol. 15, no. 5, 2004. |
| [11] | S. Ghosh-Dastidar and H. Adeli, “Spiking Neural Networks,” *International Journal of Neural Systems,* vol. 19, no. 4, pp. 295-308, 2009. |
| [12] | A. Kumar, S. Kansai and M. Hanmandlu, “Classification of different neuron behavior by designing spiking neuron model,” in *2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN)*, Tirunelveli, 2013. |
| [13] | Eugene.M.Izhikevich, “Simple Model of Spiking Neurons,” in *IEEE Transactions on Neural Networks*, 2003. |
| [14] | A. S. Cassidy and etal, “Cognitive Computing Building Block: A Versatile and Efficient,” IBM, 2013. |
| [15] | GIMP, [Online]. Available: https://docs.gimp.org/2.8/en/plug-in-convmatrix.html. [Accessed 12 2020]. |
| [16] | F. Ponulak and A. Kasinski, “Introduction to spiking neural networks: Information processing, learning and applications,” *Acta Neurobiol Exp,* vol. 71, pp. 409-433, 2011. |