The biological implementation of spiking neural network using python

By ODEMAKINDE ELISHA JESUTOFUNMI

https://github.com/elishatofunmi/pyconng2019

About Me

- I am Odemakinde Elisha Jesutofunmi
- I am a passionate developer with so much focus on achieving Intelligence on microchips.
- I am a final Year student of the Department of Electronic/Electrical Engineering (OAU).
- I love technology/Music.
- @elishatofunmi (twitter, Instagram, github, gitlab).

A quick demo

Neural Network (Definition)

A neural Network is an interconnection of neurons via synapse, built to learn patterns of an input feature (data) with the aim of having a desired output (target).

Neural networks can be:

1. Artificial Neural Networks.
2. Biologically inspired Neural Networks.

Artificial Neural Networks

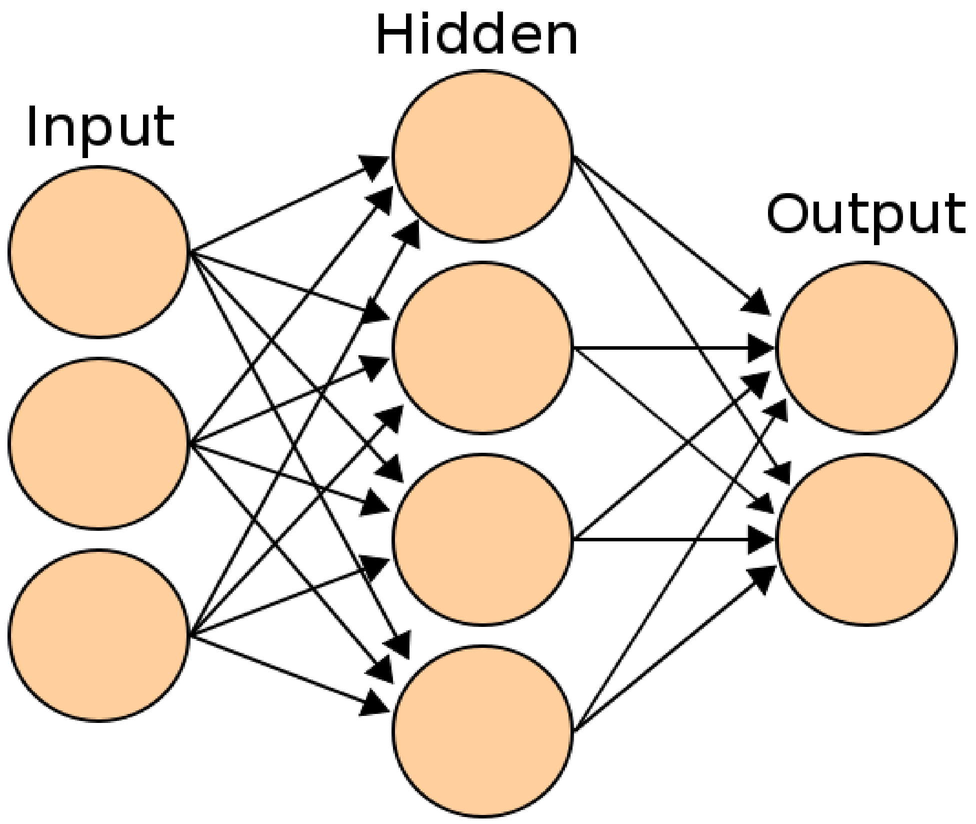
Artificial Neural Networks are interconnection of network weights and biases connected via synaptic nerves based on some certain activation function.

Biologically Inspired Neural Network

Biological inspired Neural Networks are biological-structured fully connected neuronal layers which fully mimics the human nervous system (brain inspired connectivity).

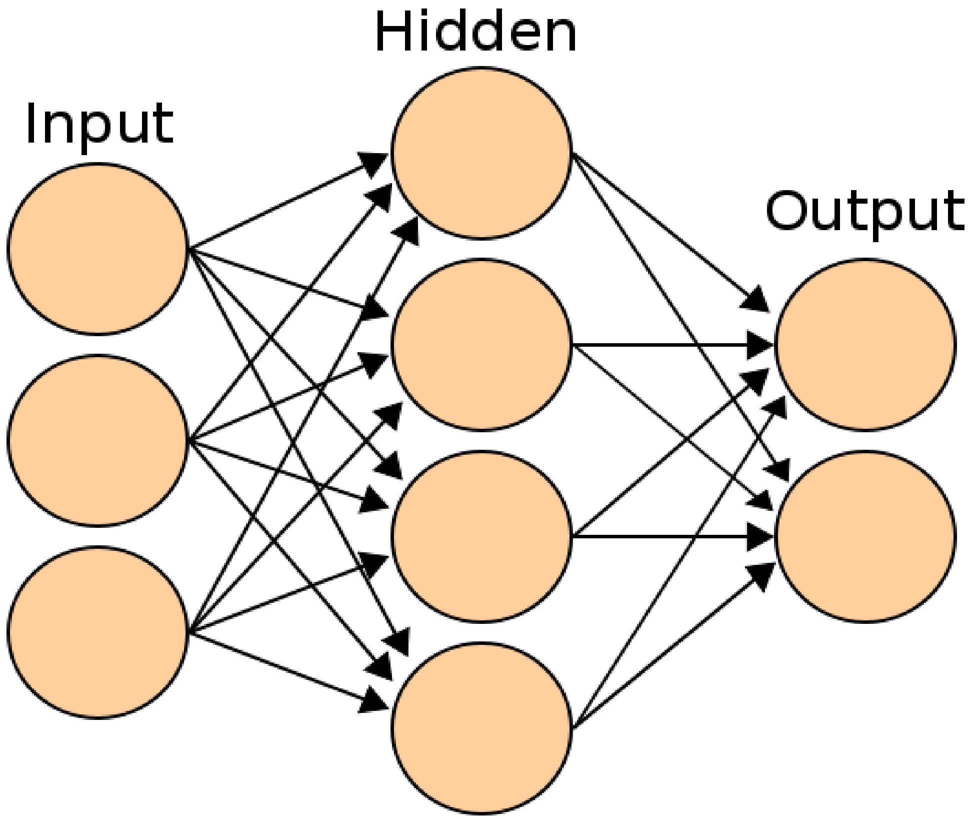
How Neural Network works

- A Neural Network consists of the input layer, hidden layer and output layer. Where the input layers are the input features, the hidden layers are the learning layers (learning all transformation and parameters to be able to replicated the input data perfectly) and the output layer is our desired results.
- Most neural network learn based on the gradient descent algorithm.



Types of Neural Network

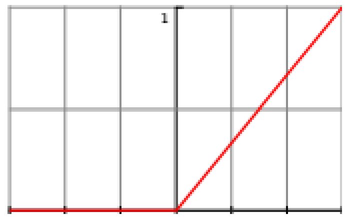
- To mention a few we have a multi-layered perceptron, convolutional neural network, recurrent neural networks, generative adversarial network, etc.



Activation Functions

In Neural Networks, Activation functions are numerical functions that computes the weights and biases begin fed from an input set of neurons to another set of neurons via synapases. Examples include, sigmoid, tanh, relu etc.

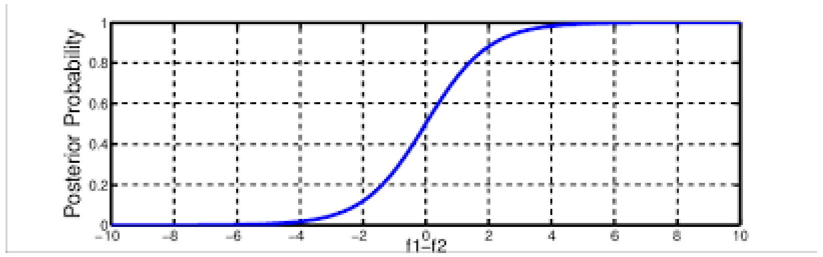
Examples of Activation functions



$$R(z) = \max(0, z)$$

Relu function

Examples of Activation functions

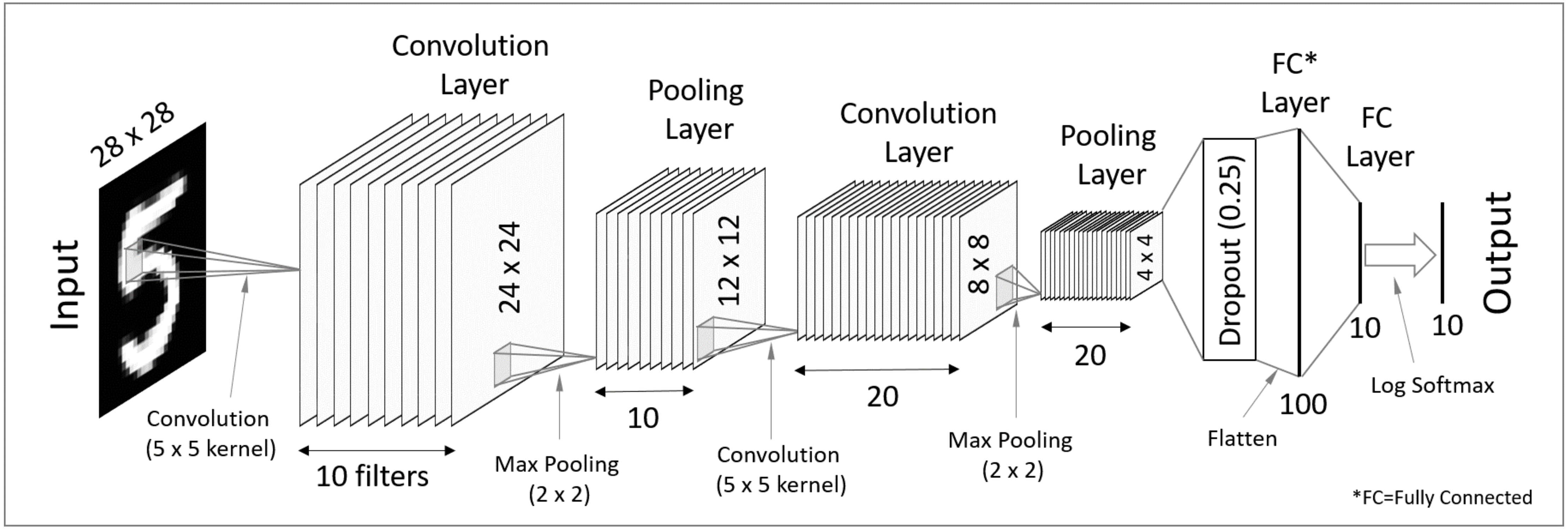


$R(z) = 1/(1 + \exp(-z))$

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

softmax function

Neural Network Examples (CNN)



What are spiking Neural Networks?

- Spiking neural networks (SNNs) are inspired form of biological neural network which are fully based on spike response of neurons to a certain learnable input features.
- They are inspired form of information processing in the human nervous system, where sparse and asynchronous binary signals are communicated and processed in a massively parallel fashion.
- It’s is mostly implemented on Neuromorphic hardware.

Why spiking neural networks?

The following outlines why Spiking neural network to should be an area of neural network African developers needs to look into:

1. It's a biological inspired form of neural network which more or less mimics the human neuronal network system.
2. It is flexible and allows for easy conversion of existing Artificial Neural networks into it's spiking equivalent using the snntoolbox toolkit (https://snntoolbox.readthedocs.io/en/latest/guide/intro.html).
3. It exhibit favorable properties in the field of neural networks like low power consumption, fast inference and event-driven information processing.
4. It could be deployed on fast and efficient Neuromorphic hardware, Field programmable gate arrays(FPGAs) etc.

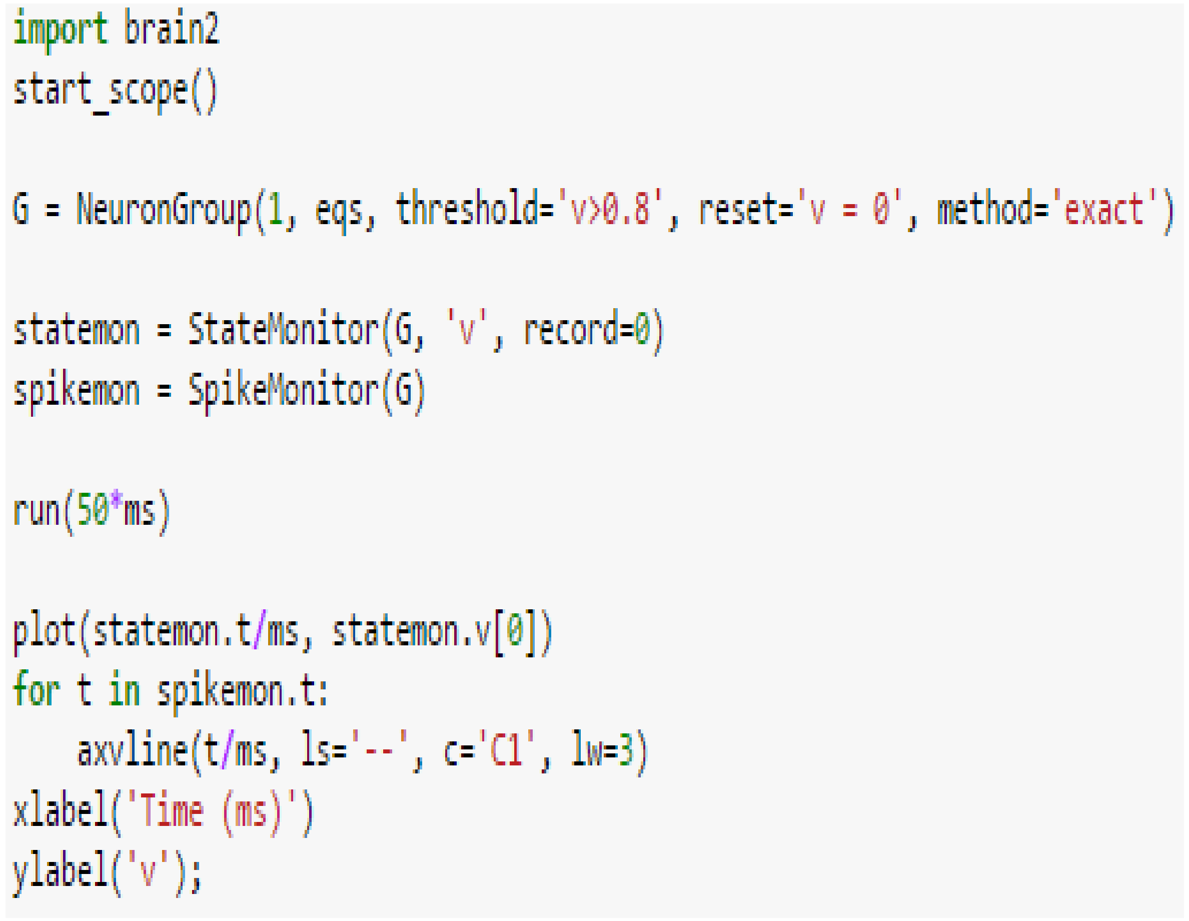
This makes it an interesting candidate for the efficient implementation of deep neural network, the method of choice for many machine learning tasks.

The Hebbian Rule

- **Hebbian learning-** states that neurons that ‘fire together, wire together’.

How spiking Neural Network works

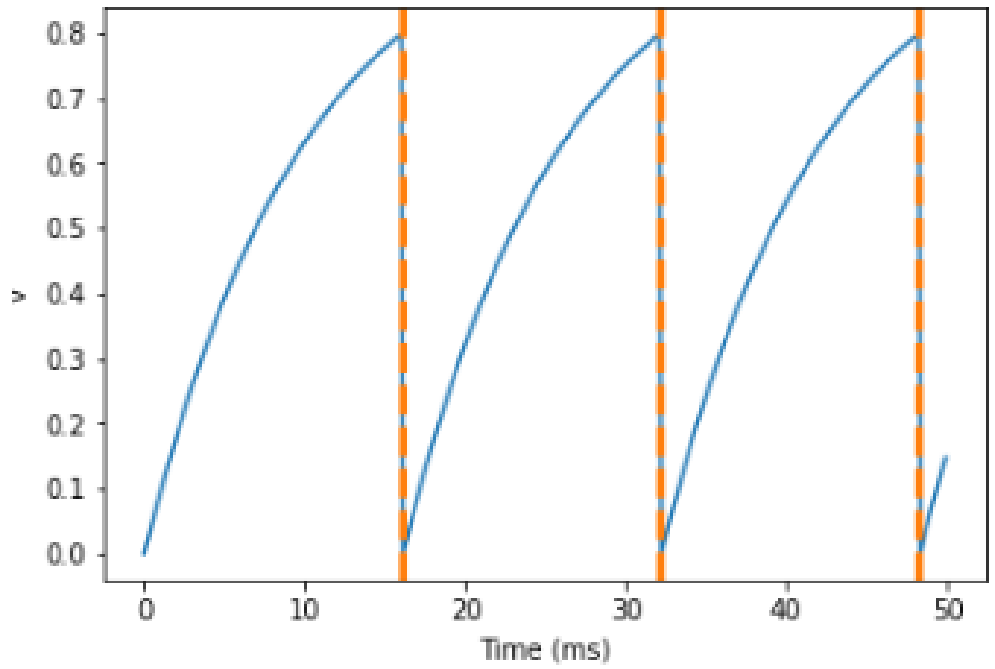
- Spikes are being generated based on the strength of the network synapses



```
import brain2
start_scope()

G = NeuronGroup(1, eqs, threshold='v>0.8', reset='v = 0', method='exact')

statemon = StateMonitor(G, 'v', record=0)
spikemon = SpikeMonitor(G)

run(50*ms)

plot(statemon.t/ms, statemon.v[0])
for t in spikemon.t:
    axvline(t/ms, ls='--', c='C1', lw=3)
xlabel('Time (ms)')
ylabel('v');
```

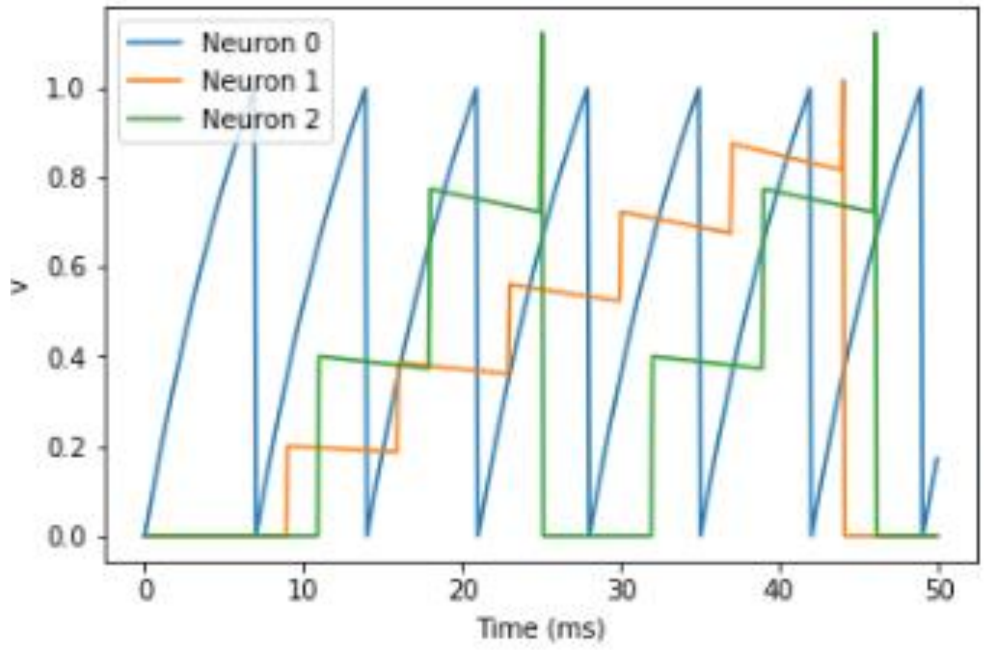
Basic understanding of spikes

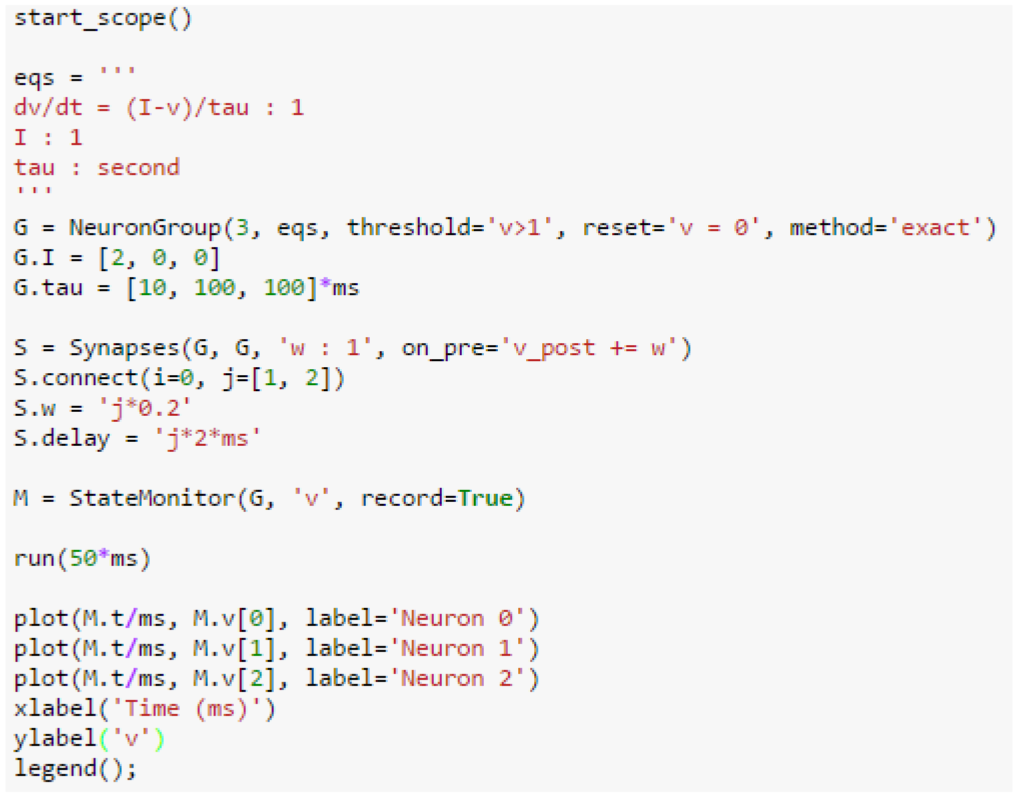
- Spikes output of each neurons.



Considering 3 neurons

- Spikes output of each neurons.

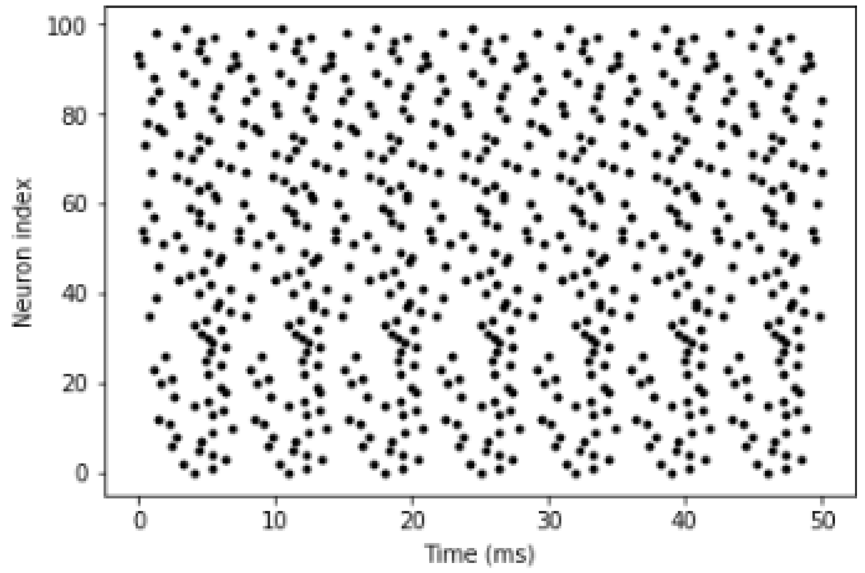


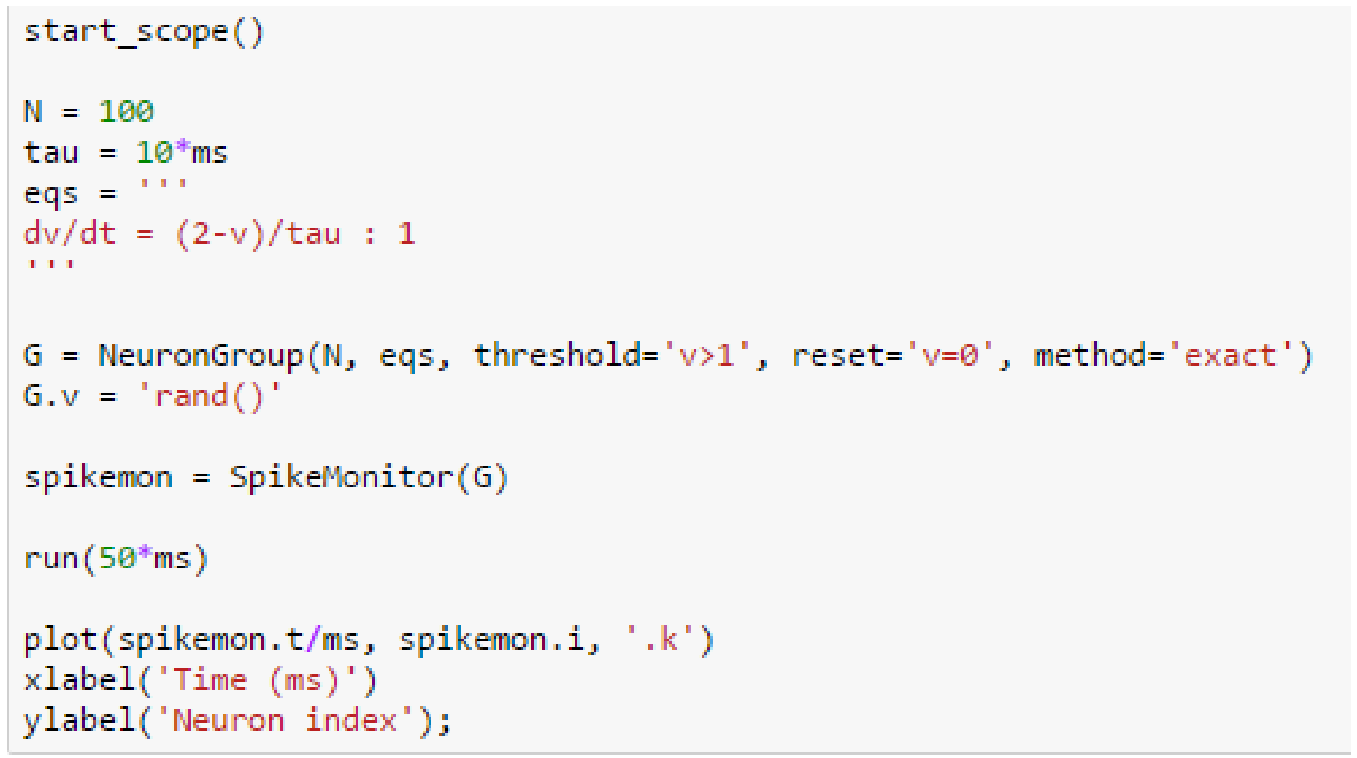


```
start_scope()

eqs = '''
dv/dt = (I-v)/tau : 1
I : 1
tau : second
'''
G = NeuronGroup(3, eqs, threshold='v>1', reset='v = 0', method='exact')
G.I = [2, 0, 0]
G.tau = [10, 100, 100]*ms

S = Synapses(G, G, 'w : 1', on_pre='v_post += w')
S.connect(i=0, j=[1, 2])
S.w = 'j*0.2'
S.delay = 'j*2*ms'

M = StateMonitor(G, 'v', record=True)

run(50*ms)

plot(M.t/ms, M.v[0], label='Neuron 0')
plot(M.t/ms, M.v[1], label='Neuron 1')
plot(M.t/ms, M.v[2], label='Neuron 2')
xlabel('Time (ms)')
ylabel('v')
legend();
```

More complex connectivity

- Spikes output of each neurons.





```
start_scope()

N = 100
tau = 10*ms
eqs = '''
dv/dt = (2-v)/tau : 1
'''

G = NeuronGroup(N, eqs, threshold='v>1', reset='v=0', method='exact')
G.v = 'rand()'

spikemon = SpikeMonitor(G)

run(50*ms)

plot(spikemon.t/ms, spikemon.i, '.k')
xlabel('Time (ms)')
ylabel('Neuron index');
```

Spiking Neural Network – Network mapping

Network Mapping: The sum of the weighted input into a neuron is mapped onto a real output value via a sigmoidal transformation-function, thus creating a graded response of a neuron to change in its input. Abstracted in this transformation-function is the idea that real neurons communicate via firing rates: the rate at which a neuron generates action potentials (spikes).

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(t_{post} - t_{pre})$$

$$W(\Delta t) = \begin{cases} A_{pre}\, e^{-\Delta t/\tau_{pre}} & \Delta t > 0 \\ A_{post}\, e^{\Delta t/\tau_{post}} & \Delta t < 0 \end{cases}$$

Spiking Neural Network – Network Mapping

When receiving an increasing number of spikes, a neuron is naturally more likely to emit an increasing number of spikes itself. As an artificial neuron models the relationship between the inputs and the output of a neuron, artificial spiking neurons describe the input in terms of single spikes, and how such input leads to the generation of output spikes.

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(t_{post} - t_{pre})$$

$$W(\Delta t) = \begin{cases} A_{pre}\, e^{-\Delta t/\tau_{pre}} & \Delta t > 0 \\ A_{post}\, e^{\Delta t/\tau_{post}} & \Delta t < 0 \end{cases}$$

Spiking Neural Network – Network Mapping

Although, Neural networks are simplified models of brains used to recognize patterns, but they waste a ton of computational power. Neuromorphic (literally, nerve-structured) systems are better at mimicking the brain, as they send stimuli to neurons in ‘bursts’ instead of a constant flow.

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(t_{post} - t_{pre})$$

$$W(\Delta t) = \begin{cases} A_{pre}\, e^{-\Delta t/\tau_{pre}} & \Delta t > 0 \\ A_{post}\, e^{\Delta t/\tau_{post}} & \Delta t < 0 \end{cases}$$

Spiking Neural Network –Network Mapping

- The idea with SNN is that only resources applied to the solution are expanded, such that non-firing neurons are suppressed. These systems wait until a neuron has reached a certain activation threshold, using a Spike-Time Dependent Plasticity (STDP) algorithm before firing.

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(t_{post} - t_{pre})$$

$$W(\Delta t) = \begin{cases} A_{pre} e^{-\Delta t/\tau_{pre}} & \Delta t > 0 \\ A_{post} e^{\Delta t/\tau_{post}} & \Delta t < 0 \end{cases}$$

Spiking Neural Network – Network mapping

- STDP is a biological process used by the brain to modify synapses, with two basic rules for molding of weights:
- A synapse that contribute to the firing of a post-synaptic neuron should be strengthened (more weight).
- A synapse that doesn't contribute to the firing of a post-synaptic neuron should be weakened (less weight).
- Multiple neurons connect to one neuron by synapse, each firing at their own rate and sending forward spikes. If a spike crosses the threshold, we monitor which pre-synaptic neurons helped it to fire. (Frederik Bussler, 2019).

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(t_{post} - t_{pre})$$

$$W(\Delta t) = \begin{cases} A_{pre}\, e^{-\Delta t/\tau_{pre}} & \Delta t > 0 \\ A_{post}\, e^{\Delta t/\tau_{post}} & \Delta t < 0 \end{cases}$$

Why you would want to try out SNN

- Its Flexible to implement and actually mimics the human nervous system biologically.
- It is fast in computation with reduced number of neuron maximization (because only synapses which contributes to output targets gets to strengthened).
- It is a feasible implementation of Neural Networks for biological computation, medical diagnosis etc.
- It runs fast and better on hardware for biological microchips.

How to get Started?

- Get to Understand how Neural Network works biologically.
- Understand the mathematical underpinning of the spike time dependent plasticity.
- Get familiar with basic understanding of SNN networks, weights and synapses using brian2, pyNN, Neuron libraries etc.
- Get familiar with neuromorphic designs of hardware microchips.
- Get robust with real life implementation using the pytorch libraries like SpykyTorch, BindsNet, snntoolbox etc.

SNN FRAMEWORKS (PYTHON IMPLEMENTATION)

SNN simulations include the following:

1. Brian 2 (written in python 2)
2. Spinnaker
3. BindsNET
4. SpykyTorch
5. PyNN etc.

Links to resources

- https://bindsnet-docs.readthedocs.io/
- https://snntoolbox.readthedocs.io/en/latest/
- https://buildmedia.readthedocs.org/media/pdf/bindsnet-docs/latest/bindsnet-docs.pdf (documentation)
- https://cnrl.ut.ac.ir/SpykeTorch/doc/intro.html
- https://www.migarage.ai/intelligence/neuromorphic-computing-tech-behind-hype/

Thank You for listening