**Introduction**

Dendrites are the branching projections of a neuron. which play an important role in integrating synaptic inputs and determining the rate of generation of action potentials by the neuron.

Dendrites are used as an antenna in neurons to receive electrical signals from other neurons. They play an important role in combining these electrical signals in neurons.

In addition to the fact that dendrites are responsible for receiving electrical signals, they can even be used as a processing system to process received signals. That is, they can be used as a neural network inside the neuron.

**Summary**

As mentioned, dendrites are a large neural network unto themselves within the neuron.

Each dendrite can process the signals received from other neurons by merging and being placed next to the soma and give it to other neurons through the axon channel.

Studying this property of dendrites can help us to have the best understanding of the human brain in the science of artificial intelligence. Additionally, we can have systems with dendritic power for information processing, disease data, and other processors.

Therefore, we should not consider dendrites only as a receptor, but can act as a neural network on its own. It can help us in the construction and design of artificial neural network with deep learning.

In this plan, we have tried to check the results of data processing with the help and implementation of dendrites on our processing system. In this code, we have simulated all the characteristics and parameters of a dendrite. We got an opinion

As we know, in order for dendrites to process information, they must be integrated with each other. After designing our model, we finally merge them.

**In general, we considered these points for simulating the system:**

1. General structure of dendrites

2. How do dendrites process?

3. What is the most important advantage of dendrite processing for neurons?

4. What are the most important processing parameters of dendrites?

**1. General structure of dendrites:**

Dendrites are extensions of the plasma rays of a branch of a neuron that transmit the electrochemical stimulation received from other neurons to the cell body or soma of the neuron from which the dendrites exit. They are often narrow and appear shorter than the axon, but tend to maintain a constant radius and can be very long.

The general structure of dendrites is used to classify neurons into multipolar, bipolar and unipolar types.

Multipolar neurons consist of an axon and many dendritic trees. Pyramidal cells are large multipolar cortical neurons with pyramidal bodies and dendrites that extend toward the cortical surface.

Bipolar neurons have two main dendrites at opposite ends of the cell body. Unipolar neurons, typical of insects, have stalks.

**2. How do dendrites process?**

In general, dendrites process two general methods, linear and non-linear. One method is for simple processing and one for complex calculations and multiple states.

Dendrites first receive incoming synaptic inputs and then integrate them with the soma. This integration of inputs is done by electrical and morphological signals. When data is absorbed by dendrites, they can be flexible. and change their original form. This makes them easy to process information. This dendritic flexibility causes a change in the size of the branching pattern of dendritic trees. By changing the size of the branching pattern, this causes movement in the dendritic morphology. Morphology plays a very important role in the processing of dendrites.

Dendrites act as a network of spikes by separating from axon exits. This neural network is called good memory computation in terms of encoding and decoding. And neural networks build memory so that they can decode the information entered into the neurons. This decoding is mostly in the form of converting electrons to biochemical and biochemical to electron conversion is done at the output.

With this situation, a dendrite acts as a processor in a nervous system that can analyze the input key from the outside and transmit it as output to other neurons.

When data enters the neural network, it is absorbed by the dendrites. After the dendritic branches find the data signals, they are deactivated. This property of dendrites allows us to see the state of integrity in the dendritic synapse. With the integration of the synaptic part, the morphology becomes active and linear or non-linear depending on the type of inputs (complexity and simplicity). This turns the dendrites into a neural network system. CPU to work

Dendrites have unique nerve impulses and action potentials, so that each dendrite can respond to any event in the body with these impulses. There is an argument that any system that can react to an action is actually a processor system. And we can see this capability in dendrites with nerve impulses.

We can think of a neuron as just a general system that is responsible for organizing the pieces of information input that enters the body. But alone, it does not have the ability to process and integrate. A neuron has a general structure consisting of dendrites, axons and soma. that each department undertakes a task. Dendrite as a branch, of course, the dendritic branch part receives data from the inputs. It is actuated by nerve impulses and processed by dendritic morphology. Then they are transmitted by the axon to other neurons.

**3. What is the most important advantage of dendrite processing for neurons?**

The most important advantage of dendritic processing for neurons is the ability to perform complex computations on synaptic inputs. Dendrites do this by receiving signals from other neurons at synapses and integrating these inputs into the dendrites. They allow the neuron to perform non-linear computations.

It is very important in processing complex information in the brain. This dendritic integration also enables neurons to recognize specific patterns of synaptic input. It underlies many cognitive functions such as learning and memory.

Through dendritic processing, neurons are able to integrate and modulate inputs, allowing them to respond differently to signals based on their time, location, and strength.

The integration of synaptic input allows neurons to perform computations such as accident detection, control acquisition, and feature selection, enabling the brain to have complex information with high capabilities.

Dendritic processing allows neurons to filter out irrelevant input and selectively respond to specific activity patterns, which increases the computational power of neural networks. Dendritic processing in a neuron leads to the generation of spikes. Dendritic (NMDA) is local. By generating these spikes, it actually makes neural output independent of the soma and provides a mechanism for local computation and flexibility.

**4. What are the most important processing parameters of dendrites?**

One of the most complex processes in the inner part of the brain and nerve cells is related to dendritic processing. Each dendritic process has important parameters. In the following, we mention the most important of these parts.

**1. Dendritic morphology:**

The shape and structure of dendrites play an important role in their ability to receive and process signals. The branching pattern and length of dendrites can affect the number and type of inputs a neuron receives.

**2. Dendritic spines:**

These small projections on dendrites are the primary carriers of excitatory synaptic input. The number, shape and size of dendritic spines can affect the strength and flexibility of synaptic connections.

**3. Dendritic ion channels:**

There are several types of ion channels, including voltage-gated sodium, potassium, and calcium channels in dendrites. These channels can influence the electrical properties of dendrites and integration of synaptic inputs.

**4. Dendritic neurotransmitter receptors:**

Dendrites contain different types of neurotransmitter receptors that can bind to different neurotransmitters. The type and number of receptors can influence the strength and characteristics of synaptic transmission.

**5. Dendritic transmission:**

Dendrites need a steady supply of proteins, organelles and other substances to maintain their structure and function. Transport of these substances inside dendrites is an important parameter that can affect their overall health and survival.

**6. Dendritic hammering:**

The ability of dendrites to change in response to damage and activity is a critical parameter in learning and memory. This can include changes in dendritic morphology, spine density, and receptor expression.

**7. Dendritic calculation:**

Dendrites can perform complex computations on synaptic inputs, such as nonlinear integration and coincidence detection.

These computational features can affect the output of neurons and the information processing capabilities of neural circuits.

**Outline for processor system design:**

In a dendritic processor system, we need to consider all the morphologies and parameters of a dendritic structure. We can also design a CPU by integrating soma and membrane potential inputs.

In this simulation, we implemented a dendritic processor system with the help of numpy, Matplotlib and seaborn libraries.

At the beginning of the work, we define a dendrite with three dendritic branches as:

Dendrite = [

[#branch 1

[.5, 1],

[1, 1.4],

[1.3, 0],

],

[#branch 2

[1.5, 6],

[1.6, 0],

[2.3, 4.8],

],

[#branch 3

[2.5, 3],

[2.6, 3],

[3.3, 4],

],

]

Here we set our dendritic branches to values. These values are the input of our data, which we considered as an array of numbers.

In order for us to start the simulation, we define the parameters required to process our system.

As we said, for dendritic processing, we must have integration with the soma. By combining this action, we actually create a neural network.

We consider a dendritic diameter. This diameter is used in the definition of synaptic responses. In this simulation, we denote the dendritic diameter by D and consider its value as 4.

D = 4 # diameter

We can set the value of Rm according to our flexibility and synaptic response. A high value of Rm will result in complex processing with high resistance and a low value will result in lower calculation power.

Rm = 100 # membrane resistance

We define the cell size for our design. For this, we must consider a certain membrane capacity. To define this parameter, we act as follows:

Cm = 100 # membrane capacitance

Considering the size of the cell, we consider the membrane capacity to be 100. It should be noted that the membrane capacity should be proportional to the cell size. Also, this membrane capacity is used to calculate the ion channel that is considered for the synaptic membrane model. It is taken, we use it.

We will have a large number of inputs going into the dendrites. Data must be integrated when entering. One of the most important parameters for this is synaptic conduction. Synaptic conductance depends on the type of synapse used. will be variable. In terms of synaptic capacity, we define the amount of synaptic conductance as follows:

g\_conductance = 5 # synaptic conductance

To perform non-linear processing or in the general case of dendritic processing, we use the inverse potential to do this. To be able to define the identity of the ions during the flow event:

E = -5# reversal potential

That the amount considered for the reversal potential should be considered just opposite to the synaptic conductance.

For the overall implementation of the dendritic processor system, we need a primary neuron. We have to do after the definition.

We consider the initial value of zero for it.

y0 = 0# initialize neuron

As said, to perform a processing in the body of the neural network, we must consider an event or an action for action. To do this, considering the spike threshold of the soma actually creates a membrane potential and forces the cell body to react. We put it.

V\_thresh = 65# spike threshold soma

Therefore, we considered the excitatory level of the membrane potential in this implementation to be the threshold level of the spike to trigger and fire the action of neuron 65.

After every possible action shot, we have to reload it like a rifle. To do this, we reset the soma potential.

V\_reset = -50# reset potential soma

In this plan, instead of 65, we considered 50-50. And this is due to many inputs with a large number of dendritic structures.

We will consider a soma timeout of 0.5 seconds for high performance of our CPU system.

dt = 0.5# time step (ms)soma

The time we consider will be relative rest for Soma. But we need to define a general rest to be able to apply complex calculations to our calculations. For this purpose, we consider a general break with a duration of 50 as follows.

t\_total = 50# total simulation time (ms) soma

We also draw a descriptive vector to show the start and end point of rest and the firing point of the soma to calculate their vector. For this, we used the Soma time vector.

t\_s = np.arange(0 , t\_total , dt)

We consider the membrane potential function as a time vector. We will consider both the starting and resting points and the firing threshold.

V = V\_reset \* np.nes(t\_s \* shape)# membrance potential vector soma

We have also considered the current time value for merging inputs with Soma.

tau = 30# membrance time constant(ms soma)

# initialize presynaptic and postsynaptic activity arrays

After we have defined the parameters and calculated the time vector and axis related to the activity of the soma during the integration of the inputs to integrate it. Now is the time to start the arrays of presynaptic and postsynaptic activity.

To define the presynaptic array, we consider dendritic flexibility and simulated soma as follows:

pre = np.random.rand(50)

In this array, we considered a range of 50 for the presynaptic array.

Then we need to consider an array for our postsynaptic activities, which we defined as follows:

post = np.random.rand(50)

That both post-synaptic and pre-synaptic are the same so that the firing and resting moments of neurons are done faster.

The two main parameters in this activity are learning mode, dendritic length and flexibility. These two parameters play an important role in the structure of dendritic learning law.

weights = np.zeros(len(pre))

alpha = 2.5# learning rate

To design and implement a dendritic processor, we need to be able to calculate the membrane potential in the soma. The main reason for calculating the membrane potential is the synaptic inputs. We have high synaptic inputs that we have been able to integrate up to this point. After integrating the data, the system should be able to calculate the membrane potential. For this, we must calculate the membrane potential as follows.

1. Considers a list of dendritic branches, each branch representing a list of synapses.
2. Synaptic response: Every synaptic response is floating. In this step, we will integrate each answer.
3. Potential: We will calculate the membrane potential in the soma.

To calculate the membrane potential, we consider a voltage. This function requires ion channel, synaptic and dendritic response parameters. The general form of our operation will be as follows:

def calculate\_membrane\_potential(dendrite , synaptic\_response , I):

potencial = 0

First, we consider a variable to store the potential and we consider the initial value 0 for it.

for branch in dendrite:

First, we consider a range to define synaptic.

for synapse in branch:

distance = synapse

We take the dendritic distance equal to the synaptic distance.

weight = distance[1]/(distance[0]+1)\*\*2

And we calculated the synaptic length as above.

potential+=weight \* synaptic\_response \* np.exp(distance\*I)

We also consider the potential for mergers.

return potential

Another parameter and calculation that will play a very important role in the dendritic structure processor is the synaptic current. Therefore, to complete the dendritic processor, we will need the synaptic current, which we consider as follows.