Spiking Neural Network Approach based on Caenorhabditis Elegans Worm for Classification

Jorge Hernandez, Karen Daza, and Hector Florez

***Abstract*—A neural network is composed of a group of neurons that make a connectome, which is a map of neural connections that allows establishing the paths between the neurons. The neural network can generate the actions of living beings with the neuronal interaction through chemical-electrical signals. The behavior of machines is not dynamic compared to the behavior of animals; then, the machine's behavior must be modeled and made by an exterior design, while in living beings, the behavior is caused by experience. Caenorhabditis Elegans is a worm model to study the connections of its neurons. In order to study the dynamic behavior in software systems based on biologic models, we created an approach to train and classify binary patterns using the structure of the Caenorhabditis Elegans' connectome. We used the connectivity of neurons of Caenorhabditis Elegans to make a custom approach to train a Spiking Neural Network using a branching factor to classify patterns instead of layers of neurons. We made a software system to show the graph of neuronal connections of the Caenorhabditis Elegans. We also used Spike-Timing-Dependent Plasticity in order to establish the strength of the weights between the connections. In addition, we used a Hodgkin-Huxley model to calculate the neuron's potential membrane and handle the spikes of the network.**

***Index Terms*—Caenorhabditis Elegans, ElegansNET, Spiking Neural Networks, Connectome, Classification patterns, Dynamic systems, Hodgkin-Huxley model.**

I. Introduction

IN software systems, the specification of the set of instructions that the system can perform must be known and the result of the execution of a source code is a specific behavior in the system. For instance, if the software system allows making the movements of a robot, the set of instructions might be the movements left and right. In this same example, the robot cannot jump because it is not designed for this functionality. In animals, the behavior depends on internal and external stimuli that trigger their behavior. Thus, the behavior in animals is not made by an outer designer because it is generated based on their experience. Then, behavior in animals is produced by the interaction of their neuronal network. Therefore, neuronal networks can help to find a solution to several computing problems by creating algorithms and approaches in the context of Artificial Intelligence (AI).

Normally, AI solves a single type of problem, in contrast to the phenomena in the real world [1]. AI has different fields such as Artificial Neural Network (ANN), Spiking Neural Network (SNN), and Machine learning (ML). Basically, an ANN has an input layer of neurons, hidden layers of neurons, and a final layer of output neurons [2]. ANNs are used in different problems to classify or predict data, so they are one of the main topics of AI [3]. In addition, they are inspired by the operations in living beings' brains, while SNNs are inspired by the information processing of living beings, where sparse and asynchronous signals are communicated and processed in a massively parallel fashion [4].

Comparing ANNs to SNNs, SNNs try to process the behavior in the most similar way to living beings using new concepts such as time, synaptic, and potential action. Thus, SNNs can help AI to solve many problems based on neuron interaction, simulating the memory and the experience of the brains of living beings.

Caenorhabditis Elegans (C. Elegans) is a worm that has different studies to understand the connections of its neurons [5], [6], [7]. The worm has 302 neurons [8] and its connectome allows showing the map of neuron connections generating a neuronal network. However, in this work, according to Hernandez et al., [9] we used 280 neurons. In addition, C. Elegans has been an animal model that has the connectome almost complete [5]. The principal movements in C. Elegans are right and left, which is very important to observe how the neural network of C. Elegans works in order to design dynamic computer systems based on the network interaction. In order to generate a solution for many computing problems based on experience and neuron interaction, it is necessary to create an approach to train an SNN.

The neural connections of C. Elegans can be represented as a directed graph. In this article, we present an approach of a software system to show the propagation of signals using an SNN to use the connections of C. Elegans. Additionally, we present an approach to train the C. Elegans neuronal network in order to solve classification problems using a Hodgkin-Huxley model, a computing-directed graph, and a Spike-Timing-Dependent Plasticity (STDP) algorithm for learning. In order to train the network of C. Elegans, we do not use layers like in an ANN; instead, we use a propagation concept selecting a starting neuron and generating the paths where the signal can be transmitted. Furthermore, we present experiments to show the SNN when the network is trained.

The article is structured as follows. Section II presents the concepts of Spiking Neuronal Networks (SNN). Section IV presents the main concepts of the C. Elegans neuronal network. Section V presents the related work focused on approaches that use C. Elegans. In sectionVI, we illustrate the proposed approach of this work. Section VII presents the software system to handle the neural network interaction. Section VIII presents the classification experiments. In section IX, we present a discussion about the main topics of the article. Finally, section X concludes the article.

Manuscript received November 6, 2021; revised August 22, 2022.

Jorge Hernandez is postgraduate student in information and communication sciences at the Universidad Distrital Francisco Jose de Caldas, Bogota, Colombia. E-mail: hrjorgee@correo.udistrital.edu.co

Karen Daza is undergraduate student in telematics engineering at the Universidad Distrital Francisco Jose de Caldas, Bogota, Colombia. E-mail: kgdazaa@correo.udistrital.edu.co

Hector Florez is Full Professor at the Universidad Distrital Francisco Jose de Caldas, Bogota, Colombia. E-mail: haflorezf@udistrital.edu.co

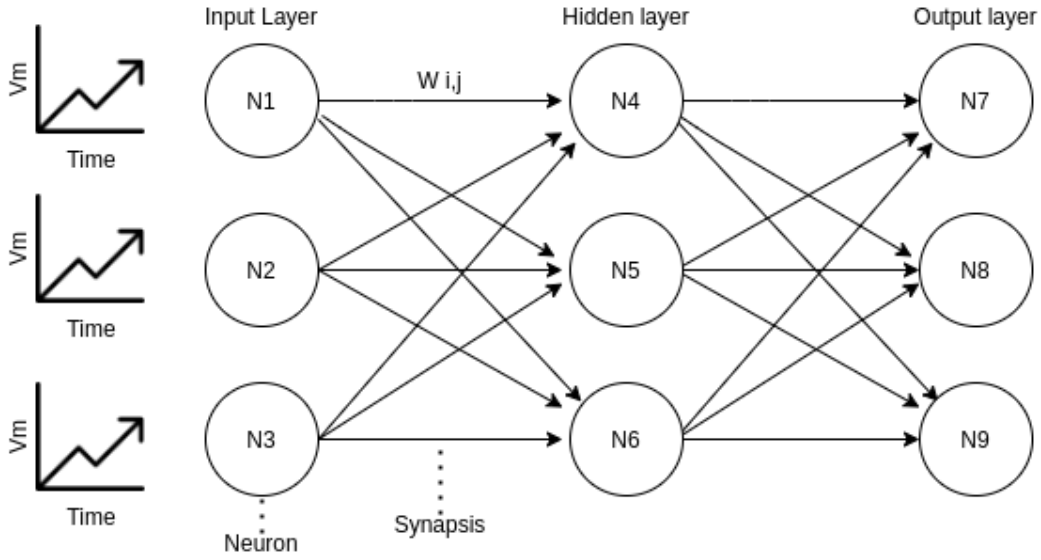

Fig. 1: **Main elements of a Spiking Neural Network:** Spiking Neural Networks must implement learning rules to handle weights based on spatio-temporal inputs. Each neuron can belong to different links that connect other neurons forming layers. The links are generated by synapses with constant communication between neurons.

II. Spiking Neural Networks (SNNs)

Spiking Neural Networks (SNNs) are characterized by their way of transmitting information since they do it in the same way as biological neurons do. To achieve this, SNNs perform a perfect synchronization of the spikes or impulses it makes during their process. Timing on this network is the main concept compared to other artificial neural networks. SNNs are the third generation of ANNs, they model the behavior of a living nervous system as it considers the spatial-temporal aspects of the input data [10].

The architecture used by SNNs is similar to ANNs' architecture, but the difference is that SNNs work with spikes. The use of applications based on biophysical models in SNN networks becomes a great challenge when implementing them due to the high computing resources required for their use. SNNs have significant potential for solving complicated time pattern recognition problems due to the inherent dynamic representation of spiking neurons [11].

Despite the limitations of these networks, they are one of the most promising networks in the future. Their learning methods are still under study because the biological model is not finished; therefore, it is not fully understood. In the future, SNNs are projected to be able to model the behavior and learning of the brain with the aim of simulating large networks in real-time with a low computing cost.

Fig. 1 presents the normal structure of an SNN. The inputs in an SNN similar to an ANN are necessary, but with the difference that in an SNN, the data represents space-time plasticity. In addition, the weights of the connections can be calculated with a learning rule. Spike Neural Networks encode information by transmitting multiple spikes or events in the form of pulse-voltage trains. From there, to pass the data to the network, the data must be encoded. Additionally, Fig 1 also presents three main layers of a neural network, which consist of the following:

- Input layer, which is responsible for receiving the data.
- Hidden layer, which is responsible for processing the information collected.
- Output layer, which is responsible to provide the processing results.

Each layer is connected through neurons and the neural network might be fully connected i.e., every neuron is connected to each neuron that belongs to the next layer as presented in Fig. 1, where the input of the network is a spike signal.

III. Neuron model

A. Hodgkin-Huxley (HH) Model

The Hodgkin-Huxley (HH) model [12] is a biological model for establishing the potential action in neuron interaction. The HH model treats the nerve axon as an electrical circuit in which the proteins are resistors and the membrane is a capacitor [13], [14]. This model can be considered to be one of the most biologically accurate spiking neuron models [15]. This model is focused on the ordinal equations presented in Equations 1 to 4.

$$\begin{aligned}\frac{dV_m}{dt} = &\frac{I}{C_m} - \frac{gk^{n^4}}{C_m}\left(V_m - V_k\right) \\ &- \frac{g_{Na}m^3h}{C_m}\left(V_m - V_{Na}\right) \\ &- \frac{gl}{C_m}\left(V_m - V_l\right)\end{aligned} \tag{1}$$

$$\frac{dn}{dt} = a_n\left(V_m\right)\left(1-n\right) - \beta_n\left(V_m\right)n \tag{2}$$

$$\frac{dm}{dt} = a_m\left(V_m\right)\left(1-m\right) - \beta_m\left(V_m\right)m \tag{3}$$

$$\frac{dh}{dt} = a_h\left(V_m\right)\left(1-h\right) - \beta_h\left(V_m\right)h \tag{4}$$

Equation 1 represents the voltage membrane that takes into consideration the external stimulus (I) and the participation of K, Na, and leakage current densities. C_m is a capacitance by unit area representing the membrane ($\mu F/cm^2$). In addition,

g_{Na}, g_K, and gl represent voltage-controlled conductance by unit area of the sodium (Na) ion channel, Potassium (K) ion-channel, and leak channel respectively ($\mu S/cm^2$). Furthermore, V_{Na}, V_k, and V_l represent the voltage gradient to the electrochemical source for sodium, potassium, and leakage current density respectively (mV).

Equations 2, 3, and 4 describe the ion-channel kinetic model by computing the derivatives of n, m, and h functions of the same variables and two voltage-dependent functions. In this case, the first term in equations is the number of closed channels that are open. The second term is the number of open channels that are close.

Table I presents the values to use a HH model.

TABLE I: Constants in the HH Model

Name	Description	Value
Cm	Membrane capacitance in $\mu F/cm^2$	1.0
g_{Na}	Sodium (Na) maximum conductances in $\mu S/cm^2$	120.0
g_K	Potassium (K) maximum conductances, in $\mu S/cm^2$	36.0
gl	Leak maximum conductances in $\mu S/cm^2$	0.3
V_{Na}	Sodium (Na) Nernst reversal potentials in mV	50.0
V_k	Potassium (K) Nernst reversal potentials in mV	-12.0
V_l	Leak Nernst reversal potentials in mV	10.613

To calculate α and β for n, m and h channels, we used the following equations recommended by the Hodgkin-Huxley model [12]. The α and β are fixed values for each ion channel and depend on the corresponding voltage. The equations to calculate α and β are Equations 5 to 10:

$$\alpha_n(V_m) = \frac{0.001 \cdot (10 - V_m)}{e^{(1.0 - 0.1\,Vm)} - 1} \tag{5}$$

$$\beta_n(V_m) = 0.125 \cdot e^{-\frac{V_m}{80}} \tag{6}$$

$$\alpha_m(V_m) = \frac{0.1 \cdot (25 - V_m)}{e^{(2.5 - 0.1\,Vm)} - 1} \tag{7}$$

$$\beta_m(V_m) = 4 \cdot e^{-\frac{V_m}{18}} \tag{8}$$

$$\alpha_h(V_m) = 0.07 \cdot e^{-\frac{V_m}{20}} \tag{9}$$

$$\beta_h(V_m) = \frac{1}{e^{(3 - 0.1V_m)+1}} \tag{10}$$

B. Model of Synaptic Plasticity

We used Spiking-Time-Dependent Plasticity (STDP) [16] to fit the weights of the network. STDP is a model that uses a Hebbian rule to establish learning by refining synapse weights during the development of learning and memory. STDP and Hebbian learning are based on biologically plausible local learning rules [17].

When a neuron receives a discrete spike, STDP is used to learn "early spike patterns" [18]. However, when the neuron receives a repetitive temporal pattern that alternates with noise, the latency between the start of the pattern and the peak of the neuron decreases during learning, so eventually the neuron never breaks out of the pattern [19].

Equation 11 presents how to update the weights of a connection. The weight change depends on the burst times between the pre-synaptic spikes and the post-synaptic spikes. The difference between the pre-times and post-times can be called activation times. The weight of the connection must be changed if the potential membrane does not trigger a threshold upon receiving a given spatio-temporal pattern.

$$W(\Delta t) = \begin{cases} +A \cdot exp(-\Delta t/\tau+), & \text{if } \Delta t \geq 0 \\ -A \cdot exp(\Delta t/\tau-), & \text{if } \Delta t < 0 \end{cases} \tag{11}$$

Where:

- Δt is a time difference between pre-synaptic and post-synaptic neuron spike times.
- $+A$ is a coefficient for potentiation.
- $-A$ is a coefficient for depression.
- $\tau+$ and $\tau-$ are a time constants.

C. Data encoding

In the process of data encoding, population encoding is used. This is one of the techniques most commonly implemented when analyzing SNNs. This technique receives a data set with real values, which are converted into a sequence of spikes. This conversion process is performed through multiple Gaussian receptive fields. Each x_i of X the entries is independently encoded by a group of P one-dimensional receptive fields. The Gaussian receptive field for a value is calculated depending on, σ_i as presented in Equations 12 to 14.

$$g_i(x) = \frac{1}{\sigma_i\sqrt{2\pi}} exp\left(\frac{(x - \mu_i)^2}{\sigma_i^2}\right) \tag{12}$$

$$\mu_i = I_{min} + \frac{(2f-3)}{2}\frac{(I_{max} - I_{min})}{P-2}, f\epsilon\{1, ..., P\} \tag{13}$$

$$\sigma_i = \frac{1}{\beta}\frac{(I_{max} - I_{min})}{P-2}, f\epsilon\{1, ..., P\} \tag{14}$$

Where:

- I_{min} is the minimum value of the input. $I_{min} = -1.5$.
- I_{max} is the maximum value of the input. $I_{max} = 1.5$.
- β controls the width of each Gaussian receptive field, in this case $\beta = 2$.
- P is the number of receptive fields used for encoding, in this case $P = 5$.
- μ_i is the center of the receptive field.
- σ_i is the width of the receptive field.
- g_i is a Gaussian peak.

IV. Neuronal Network Structure of C. Elegans

The neural network of C. Elegans is composed of 6260 connections of neurons [9]. Large populations of neurons have the ability to carry out multiple complex parallel processes provided by the highly ordered architecture of the network [20]. When neurons are stimulated by external sources, two types of responses take place: passive and active [21].

The types of neurons in C. Elegans are the following:

- **Sensory (SN)**: It is a type of neuron that responds to the conditions of its surroundings or environment.
- **Motor (MN)**: This neuron is in charge of transporting the information received from the central nervous system to the muscles.
- **Inter (IN)**: This neuron can interconnect with other neurons in its space, allowing communication between sensory neurons and motor neurons.

The signal flow when an external input is received is:

- First, the neuron sensory receives the signal.
- Second, the signal is passed to the interneuron and the inter passes the input to the motor neuron.
- Third, the motor neuron sends the signal to muscles to C. Elegans movement (right or left).

Understanding the network of neuronal connections in the brain is necessary to unravel the way it works and processes information [22]. In this way, neuronal connections are an essential part to understand how the brain works. With this in mind, we seek to understand in a correct way the functioning of the brain. Neurons are usually grouped by their location or by the type of connections they establish between them.

The connections that neurons make to each other is called a synapse, which is a basic system where a group of neurons can communicate with each other. Through this system, neurons receive and distribute information, and this information is expanded in the form of electrical impulses from one neuron to another. This is how brain activity is developed. For the C. Elegans connectome, its synapse is too small, so it must be evaluated with a specialized electron microscope. In addition, when evaluating both sexes of C. Elegans, the connections are different for each one because some of these connections vary in their strength.

The structure of this network can be represented by means of a graph, where each node of the graph represents a neuron in the network. Its arcs, also called edges, resemble the connections that exist between the group of neurons.

V. Related work

In this section, we present some works that tried to show the interaction of the neural network of C. Elegans with computing or biophysics approaches. Some works do not directly use C. Elegans neurons, but we consider them very important to be included in this section in order to show works that try to represent the neuronal stimuli and signal propagation between neurons.

Brader et al., [23] present a spike-driven model in order to classify patterns in a semi-supervised way. Also, the authors are focused on showing how to save the information in the memory encoding labels. The authors aim to make hardware to classify stimuli. The dataset used is a binary representation of characters. The authors showed that a simple network of integrate-and-fire neurons connected by bi-stable synapses can learn to classify complex patterns.

Galluccio et al., [21] present a hybrid (molecular and electromagnetic) model to allow communication between neurons. This approach includes blocks to propagate the signal. The blocks allow the transmitter (TX) to send an electrical signal through a channel to the receiver (RX). The transmitter and the receiver are defined as pre-synaptic and post-synaptic elements. The concept of the potential action is also used. The authors conclude that the proposed model can help in the future to communicate the nervous pulses in the human body with a nanomachine in order to replace missing parts of the human body.

Kim et al., [8] designed a software system to represent the interaction between neurons of C. Elegans to allow the stimulation of the neurons. The authors discuss that the work can help to study the dynamics and the structure of the neuronal system of C. Elegans. The main components in this approach are the visual interface and neuronal integrator. This solution can allow updating the connection properties of neurons. The visual interface shows the connections and marks by the colors of the neuron types (sensory, inter, and motor). Based on this, the authors might understand how to translate the neuronal activity in C. Elegan's behavior.

Wicks et al., [24] present a novel strategy to predict the polar configuration, which in this case corresponds to the excitatory and inhibitory connections. The authors use neurons of C. Elegans to make four experiments to predict the polarities of seven of the nine-cell classes of the tap withdrawal circuit.

Bhuiyan et al., [15] present the analysis and implementation of the following two spike-based neural models i.e., Izhikevich and Hodgkin Huxley. The objective of using the models is to enable the recognition of a number of 48 images previously defined by the authors. Both models were trained with the 48 images plus several images with distortion. The two models are the most accurate from the biological point of view. The authors studied the idea that these models allow modeling the visual cortex of an animal or a person. The authors suggest that the Izhikevich model is the best candidate for implementing a large-scale visual cortex model.

Wang et al., [25] propose the development of a neural network model based on peaks to enable decisions making processes. The proposed model is applied to a game called *Flappy Bird*. With this information and after several pieces of training, the model manages to obtain a learning capacity very similar to humans. The developed network is designed to play strategy games, where the network does not know anything about the game, but after several pieces of training, it begins to develop a capacity for understanding. The designed network is based on two modules: environment perception and autonomous learning.

Wang et al., [17] discuss the creation of a network with an adaptive structure in the learning process, where certain nodes are pruned depending on the activation. In comparison with our approach, our network does not use a pruning method because the network trains the chosen nodes using a propagation concept.

VI. Proposed approach to the use of C. Elegans connectome

This section presents our approach for using SNNs based on the structure of C. Elegans in order to train the network using the HH model and STDP. The net was named *ElegansNET*. We generate a directed graph to represent the neuronal connections. The neuronal network (M) is composed of a set of neurons (B) and a set of links (L).

Then, let $\{b_1, b_2, ...b_n\} \in B$, where $n = 280$, if $B = \emptyset$, then it is necessary to populate the nodes in B. When $B \neq \emptyset$ the training in the network can be applied to generate

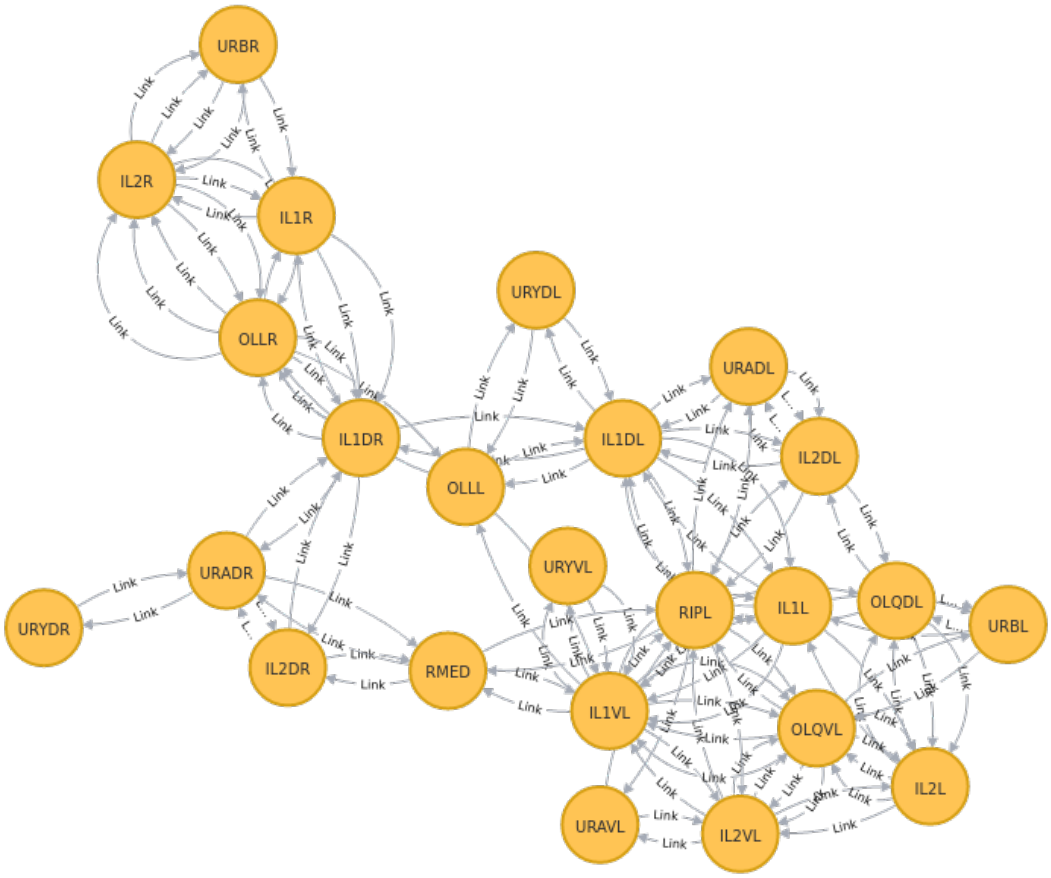

Fig. 2: **Directed graph of the approach:** The graph is stored in a Neo4j database. The yellow circles represent neurons and the arrows represent synapse connections between neurons. Each connection is composed of a weight corresponding to pre-synaptic and post-synaptic neurons. For each neuron, the voltage history, membrane potential, neuron type, and name are stored. The graph is composed of 6260 connections and 280 neurons.

learning. In M, the neurons can be connected between them through a link ($L_j \in L$). One link L_j has an initial node (a) and terminal node (b). Thus, the directed graph is $M = (B, L)$, where $|B| = 280$ and $|L| = 6260$. The network meets the following requirements:

- $M \neq \emptyset$
- $L \subseteq \{(a, b) \in BxB : a \neq b\}$
- Each link have a weight $w_{i,j}$

Fig. 2 presents a directed graph that represents a network graph [26], where the yellow circles are neurons (B_i) with their neuron names, as well as the arrows represent the connections (L) between neurons. The network graph is stored and handled on the graph database management system called *Neo4j* (https://neo4j.com/). Every connection has a direction representing the starting neuron, which is pre-synaptic, and the ending neuron, which is post-synaptic.

A. Network Architecture

The network has been designed like a directed graph, so each neuron is connected to another neuron. Fig. 3 shows the network structure when the learning rule is applied. When the learning process is performed, an initial neuron is selected and the propagation algorithm calculates the paths where the signal of voltage must be transmitted. Additionally, when the initial neuron is chosen, it is necessary a neuronal interaction between the inhibit neurons (in this case, inhibit neurons do not establish a reaction against the signal) and excitatory neurons. This intereaction generates the paths where the signal can pass. Finally, the paths get a resulting neuron in order to choose the output class with the weight's ponderation of the paths.

The graph paths are composed of excitatory neurons, in this case, sensory or motor neurons. The paths also are composed of the inhibiting neurons with an interneuron type. The receiving neuron eventually adds up the afferent stimuli, and when more excitatory signals are received, the neuron fires and sends signals to other neurons. If the sum of the signals is inhibitory, the neuron does not fire and does not influence the activity of other neurons.

B. Network Populating

In order to generate the computing neuronal network, it is necessary to populate the neuron's data and the connectivity relations to establish the structure of the network. Table II presents the principal attributes for each node of M. The main attributes are the neuron name (e.g., AVR according to Brenner [5] conventions for the neurons names), history voltage, membrane potential value, and type of neuron.

TABLE II: Node data

Attributes	value
Neuron Name	Text according to Brenner
Membrane potential	Numeric value
Type neuron	Inter, Motor, and Sensory

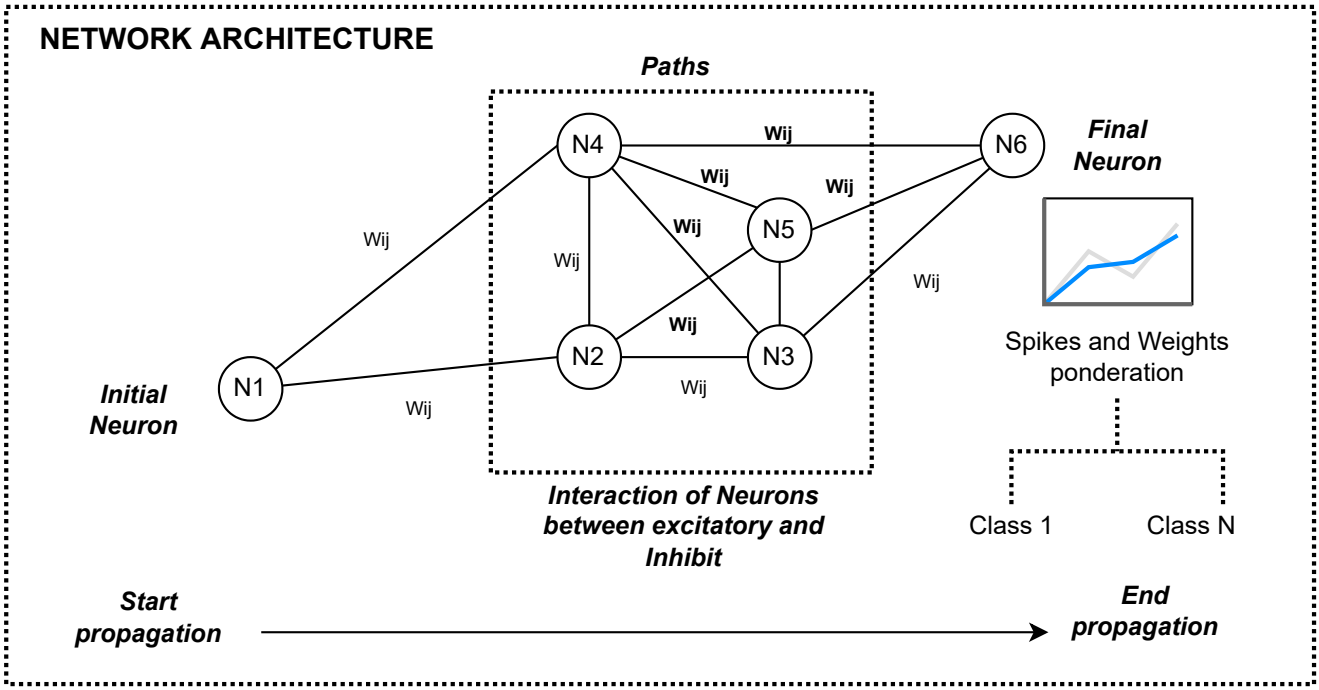

Fig. 3: **Network Architecture:**. The signal is propagated through different pathways. It is done by activation or deactivation of the post-synaptic neuron. A synaptic neuron can send different impulses depending on the structure of the connection. These connections of C. Elegans have been studied before; however, we are applying a learning rule to the neuronal connections and adding a value for the weight of each connection.

Algorithm 1 Network Populating

Require: $Links(L)$ ▷ dataset
Require: $pre - synapticnodes(P_R)$ ▷ from L
Require: $post - synapticnodes(P_O)$ ▷ from L
  $B = P_R \cup P_O$ ▷ Extract the neurons
  Save elements of B in DB
  $currentConnections \leftarrow fromDB$
  if $CurrentConnections$ are empty **then**
    while $nextConnection$ from $connections$ **do**
      $neuronpre - synaptic \leftarrow nextConnection$
      $neuronpost - synaptic \leftarrow nextConnection$
      if $neuronpost - synaptic$ is not in graph or $neuronpre - synaptic$ is not in a graph **then**
        insert PostNeuron or PreNeuron
      end if
      if $neuronpre - synaptic$ is not linked to $neuronpost - synaptic$ **then**
        create Link connection a pre and post neuron
      end if
    end while
  end if

In this work, we use a dataset presented in [9] to analyze the neuron connectivity network of C. Elegans and generate weights for each neuron connection in order to create a directed graph. In this case, the nodes of the graph are the neurons. For each link in the network graph, we use the Equation 15 to establish the first weights. [9]:

$$wi_j = 0.2 * cluster\_score + \frac{T_n}{T_f + NBR} \quad (15)$$

Where NBR is the number of synapses, T_n is the number of occurrences of the neuron in the connections, and T_f is the total number of neurons in the network.

Algorithm 1 presents the steps to populate the network when neurons or links do not exist in the directed graph. To run the algorithm, it is necessary to use the dataset of connections L from [9]. In the dataset of connections, each connection has a pre-synaptic and post-synaptic neuron. Then, the goal is to extract all neurons and save unique neurons in the directed graph. Thus, the connections for the nodes can be created. If a neuron is not present in the directed graph, it is added. If the connection is not yet present, the relationship between the pre-synaptic and post-synaptic neurons is added.

Algorithm 2 Network Training

Require: $numberOfConnections$ ▷ Number
  $epochs \leftarrow numberOfConnections$
  $iteration \leftarrow 0$
  while $x_i \leftarrow X$ **do** ▷ Encode the entries from real data to firing times
    while $receptiveField$ **do**
      $x_i \leftarrow calculatetheGaussianreceptiveoutput$ ▷ According to equation 12
    end while
  end while
  while $iteration <= epochs$ **do**
    $pre - synapticNueron \leftarrow currentConnection$
    select a sensory neuron
    $paths \leftarrow$ FINDINGPATHS($sensoryNeuron$) ▷ from Algorithm 3
    while $nextPath \leftarrow paths$ **do**
      Calculate the membrane potential of the pre and post
      check if spikes are not in a refractory times
      update weight of synapse ▷ According to equation 11
    end while
    $iteration \leftarrow (iteration + 1)$
  end while

Algorithm 3 Finding Paths for signal propagation

Require: $neuronStarted$
  function SEARCHBRANCH($neuronStarted$)
    Search the connections where neuronStarted is pre
    return finding connections
  end function
  $paths \leftarrow$ SEARCHBRANCH($neuronStarted$)
  while $nextPath \leftarrow paths$ **do** ▷ Calculate the subpaths
    Obtain the pre-synaptic neuron
    $subpaths \leftarrow$ SEARCHBRANCH(pre)
  end while
  Return the subpaths and principal paths

C. Network Learning

We use the STDP algorithm in order to train the network to adjust the weights according to the potential membrane calculated with the HH model. We did not use layers, but we used a path-branch concept because we generated a directed graph with the neurons of C. Elegans. In an experimental way, we observe that from a neuron that sends a signal (pre-synaptic) to connected neurons (post-synaptic neurons), the post-synaptic neuron can propagate the signal to another connected neuron. In this case, depending on the direction of the connection, a neuron can be set like a pre-synaptic and then a post-synaptic.

Algorithm 2 presents the steps to train the network ElegansNET. For each epoch, an initial neuron is chosen to calculate the paths or connections. From the connections of the neurons, each connection is visited calculating the membrane potential of the pre-synaptic and post-synaptic neurons. Then, the weights are changed using the difference in the firing time between the pre-synaptic and post-synaptic neurons.

D. Propagation of signals

Since the network is a graph, we use a finding paths algorithm to search the pre-synaptic and post-synaptic neurons and then generate a signal propagation between the neuron sender and neuron receiver. To propagate the values, it is possible to go through the list of connections of the C. Elegans neurons. In each propagation, the path would be the same and the interaction of the network would have the same effect between its nodes. The goal is to carry out a propagation start according to the pre-synaptic neuron from where the signal starts. Algorithm 3 is used to generate the paths from a pre-synaptic neuron, searching all possible paths where a signal must be transmitted.

E. Network Classification

ElegansNET uses the adjusted weights of the algorithm. Assuming that the optimal weights of the network are obtained using the STDP algorithm and the HH model, the weights are used to classify the input patterns into different classes. The ElegansNET classifier uses the *winner takes all* strategy to suppress non-firing neurons and produce distinguishable results.

The main steps in pattern classification are:

- Finding the propagation paths.
- For each neuron (pre-synaptic and post-synaptic) of each propagation connection, the membrane potential is calculated.

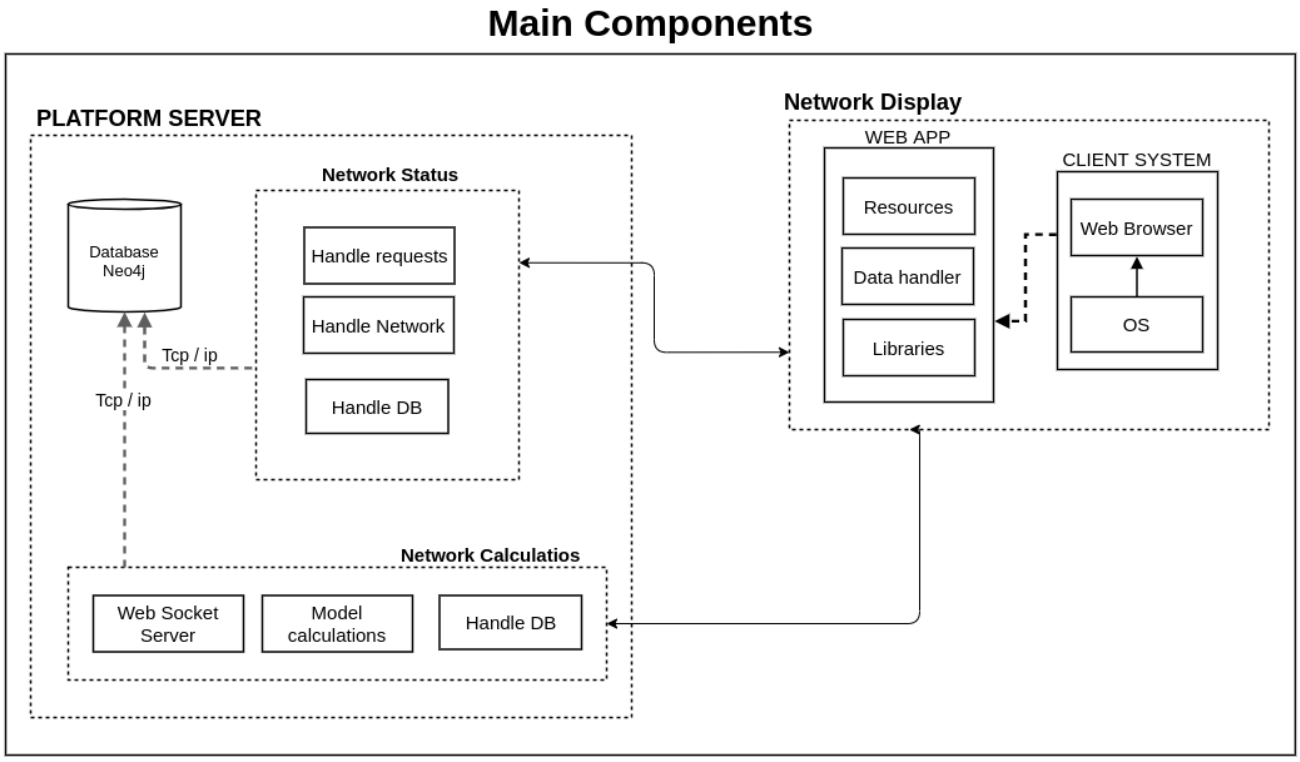

Fig. 4: **Main components in the software system:** This system is integrated by three components, where each component is in charge of performing a specific function. The first component is *Network Display*, which is in charge of processing the web application that allows the visualization and manipulation of the neural network. The second component is *Network Status*, which is in charge of executing the required web services, which enables returning the requested data to the neural network to be stored in a database. Finally, the third component is *Network Calculations*, which is in charge of creating a web socket to generate the propagation signals from the previously obtained data.

- An encoding of the incoming data is generated at a peak frequency proportional to the membrane potential.
- For each x_i of X, at each time step, the potential of the neuron is updated according to the input peak and the associated weights.
- The first output neuron performs an inhibition on the rest of the output neurons.
- The classifier counts the number of peaks to verify the output class.

VII. Software System

We developed a software system to handle the network interaction and to show the network neuronal graph. The software system allows generating a propagation of an electrical signal between a pre-synaptic neuron and the post-synaptic neuron. Fig. 4 presents the main components of the software system that are explained as follows:

- **Network Status**. This component is a Python app that runs on a server. The purpose of this component is to create a web service to return the data about the neuronal network. The data is stored in a Neo4j database. The main entities in the graph database are the nodes (neurons) and the links between the nodes.
- **Network Calculations**. This component is a Python app. The main purpose of this component is to create a web socket to receive a message about the signal propagation from a *Network Status* and to make the necessary calculations to generate a propagation signal in order to save the data in the Neo4j database. Then, the component notifies that the network has been changed through broadcast mode.
- **Network Display**. This component is a React application. The main purpose is to show a graphical interface to the user to get the data from the *Network Status* component, and to receive-send messages to the *Network Calculations* component to generate new changes in the network. The network is shown to the user in a graphical way, which classifies the neuron types by color.

The *Network System* component and *Network Calculations* component are back-end components. The *Network Display* component is a front-end component. The calculations for the network are not performed in the *Network Status* component because the propagation of the network can take a long time. Since the calculations are asynchronous, we separated the calculations in order to focus only on the propagation and training calculations.

Fig. 5 presents a Business Process Modeling Notation (BPMN) model to illustrate the flow to generate changes in the neuronal network. In the software system, the interaction is composed of the components (*Network Status*, *Network Calculations*, and *Network Display*) already explained. After loading the resources of the web application or *Network Display* component, the first action is getting the network data from the *Network Status* component. The network data can be a first network state (without any signal generation)

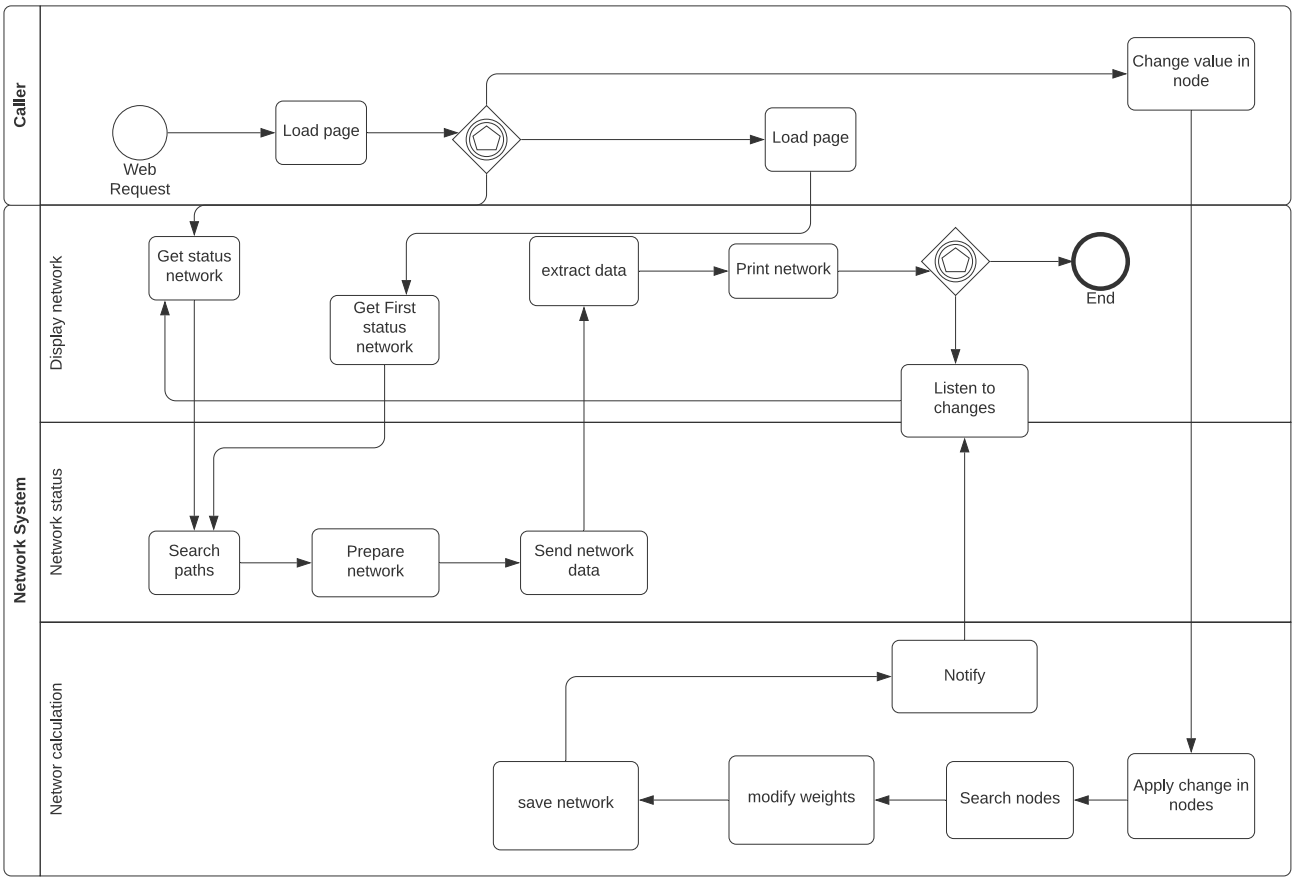

Fig. 5: **BPMN (Business Process Modeling Notation):** This BPMN model is in charge of presenting the flow and activities that the system must follow to carry out the processes and calculations previously defined by the network. In addition, it presents the transition made between each of the components of the software system, which are *Network Display*, *Network Status*, and *Network Calculation*. In each component, the functions performed are deployed until the final result is presented to the user.

or the current network state (with changes saved in the network). When the *Network Display* component gets the data on the network from the *Network Status* component, the *Network Display* component extracts the data and prints the network to the user interaction.

When the network is deployed to the user, the user may try to change the value of the signal for two neurons. Then, the *Network Display* component sends the inputs to the *Network Calculations* component. Later on, the *Network Calculations* component searches the paths and generates the calculations. Then, it notifies the web application indicating the calculations are finished. Finally, the *Network Display* component can get the current state from the *Network Status* component.

Fig. 6 presents a screenshot of the graphical interface generated in the *Network Display* component to view the interactive system that presents an animation of the neuronal network of C. Elegans. Using this graphical interface, the user can view all neurons and its connections by type of neuron (inter, motor, sensory). The user can click on a neuron to show its information. Also, the user can search for a pre-synaptic neuron and generate a signal propagation to a post-synaptic neuron in order to generate the paths and update the weights for each connection.

VIII. Classification Experiment

We performed several experiments to calculate the accuracy of a classification problem and to determine whether the approach is valid to classify patterns. The idea in the classification problem is to separate the classes with a previous training algorithm and then generate output-based data.

A. Dataset

We used a typical data set for the classification technique. The data set presents 4 attributes and the target class. The objective of the dataset is to identify an iris flower [27]. The attributes are petal length, petal width, sepal length, sepal width, and flower species. The last attribute is the class or output target in the dataset.

In addition, this dataset contains three species of iris, which are Setosa, Virginia, and Versicolor. For each of these species, there are 50 samples, which are analyzed with the parameters already mentioned. This dataset is open access and is in a special repository of machine learning.

B. Results

1) Propagation Algorithm: Algorithm 3 was proposed to be able to obtain the paths starting from a point of origin

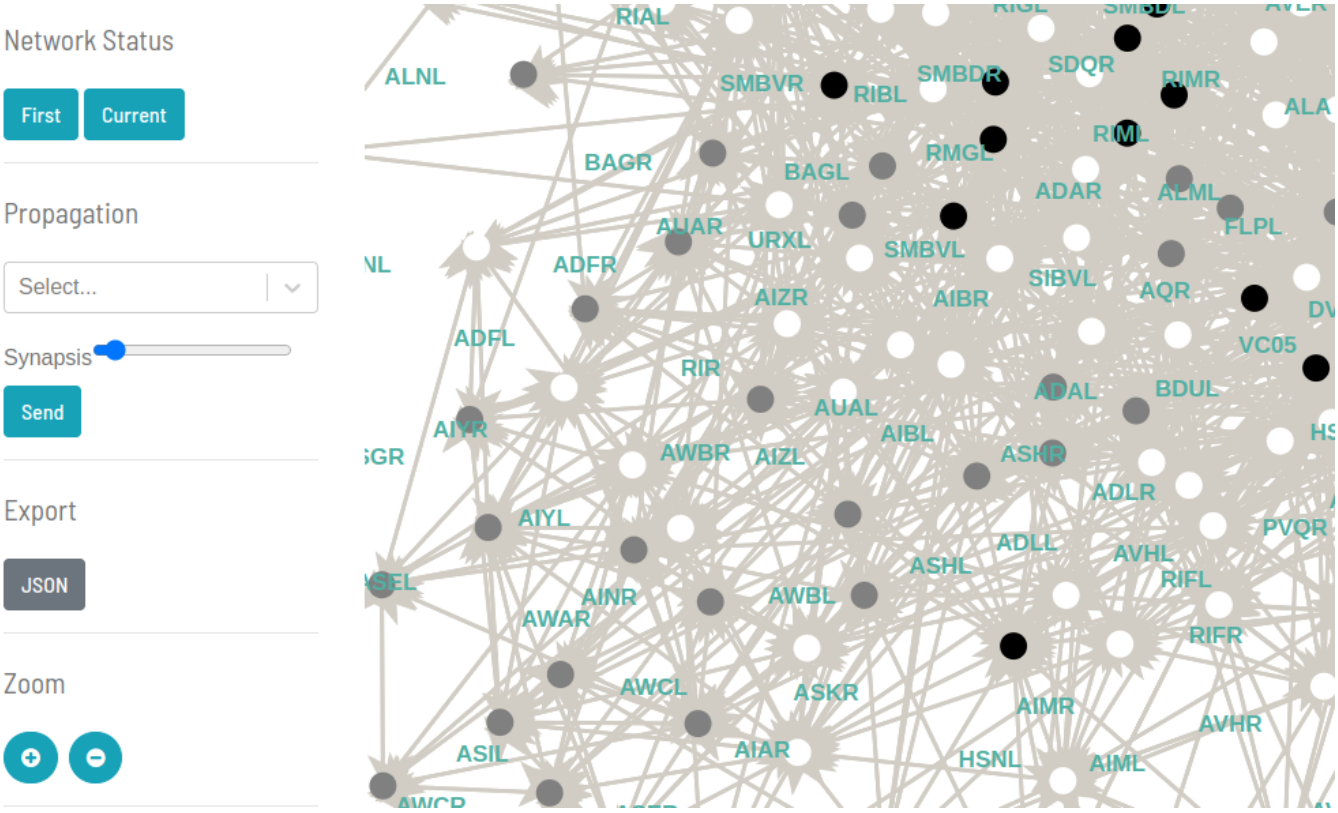

Fig. 6: **Screenshot of the Graphical interface (Network status component):** The system allows showing the structure of the network to the user. The user can show the relations of neurons by the type of neurons. The user can download the network structure in JSON format. The user can modify the weight of one relation by selecting a pre-synaptic neuron. Then, the network sends the signal to post-synaptic neurons using our propagation concept.

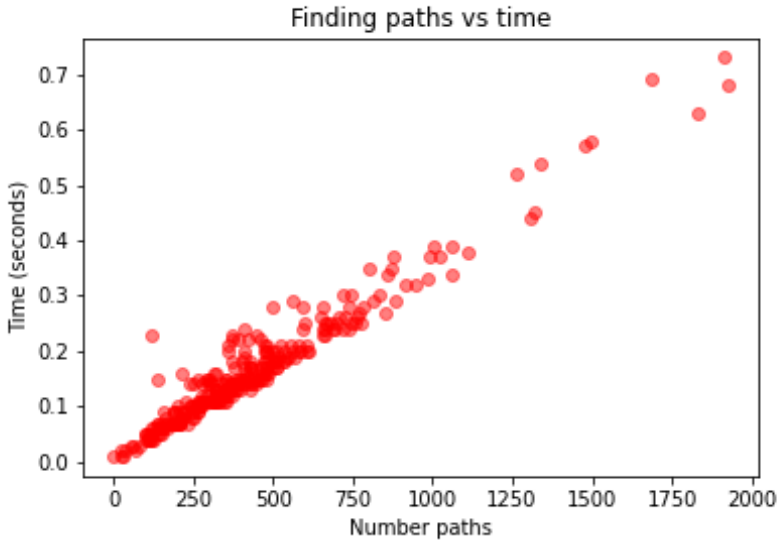

Fig. 7: **Results for propagation algorithm:** Time vs number paths in the experiment for each neuron using Algorithm 3.

that in this case, corresponds to an initial neuron. In this way, this algorithm begins searching the connections where this initial neuron is pre-synaptic, then recursively searches those paths for the connections where in each sub-connection the post-synaptic neuron becomes pre-synaptic to send the signal. In this way, a sort of subnet is created. The algorithm was tested through an experiment, where for each of the 280 neurons (regardless of the type of neuron i.e., sensory, inter, or motor), the necessary paths were calculated to propagate a signal starting from such point of origin. Additionally, the time to compute the paths from each initial neuron was registered.

Fig. 7 and Fig. 8 present the results of the experiment. Fig. 7 presents the times registered when taking each neuron as the point of origin. Thus, the algorithm presents a good result in comparison with the number of paths taken in the experiment, since the time that the algorithm takes is proportional to the number of paths found for the origin

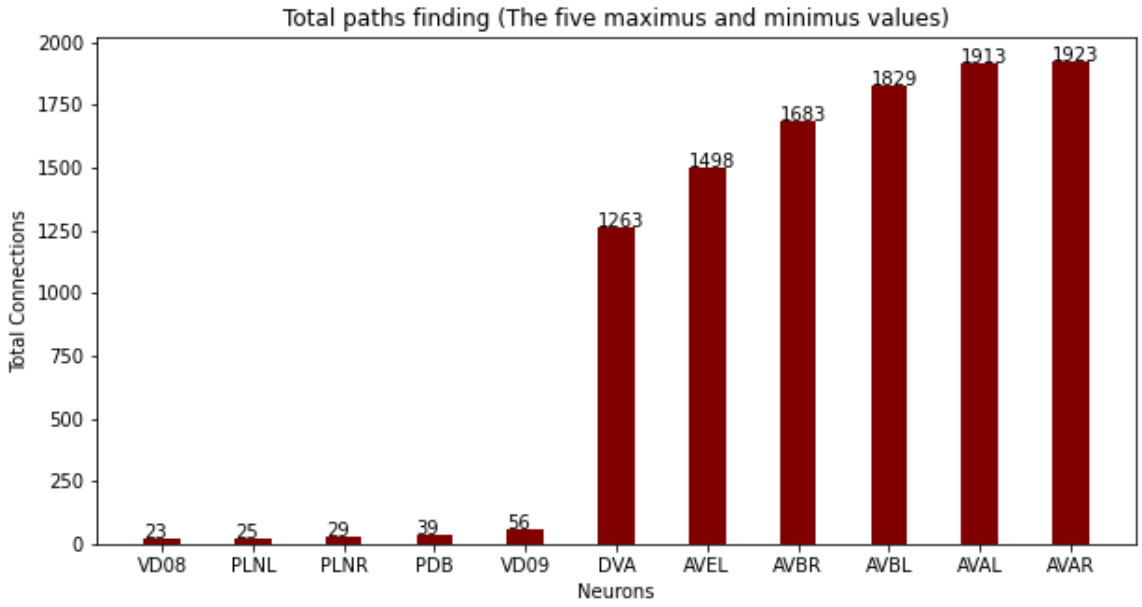

Fig. 8: **Results for propagation algorithm:** Maximum and minimum total connections in the network calculated using Algorithm 3.

neuron. The algorithm lasts less than a second, even on the neurons where the algorithm must evaluate the largest paths, which demonstrates the algorithm's efficiency and effectiveness.

Fig. 8 is a bar chart to represent the 5 neurons with the lowest amount of connection and the 5 neurons with the highest amount of connection. In this way, the maximum amount of connections oscillates between 1498 and 1923 for the *AVEL* and *AVAR* neurons respectively. The minimum amount of connections is 23 for the neuron *VD08*. The types of neurons included in the bar chart are:

- *VD08*: Motor neuron
- *PLNL*: Sensory neuron
- *PLNR*: Sensory neuron
- *PDB*: Motor neuron
- *VD09*: Motor neuron
- *DVA*: Inter neuron
- *AVEL*: Inter neuron
- *AVBR*: Inter neuron
- *AVBL*: Inter neuron
- *AVAL*: Inter neuron
- *AVAR*: Inter neuron

In addition, this experiment resulted in the neuron *VC06* having no paths when applying the algorithm. This has also been confirmed in [28], [29], where it is stated that this neuron has no associated connections or synapses.

2) Training: The neural network was trained with the iris flower data set. We carried out training with 100 epochs for the network. Fig. 9 presents the accuracy result for each epoch for the testing and training phases. Analyzing each epoch, the major difference between the testing and training was in epoch number 13 with a difference of 54.9%. The maximum accuracy was 78% for the training phase and 82% for the testing phase. The minimum accuracy was 36% for the training phase and 33% for the testing phase. With these results, there is an increase in the network accuracy after the execution of each epoch, despite the accuracy may decrease at certain epochs. In the first 35 epochs, the drop in the accuracy could occur because the peaks can be classified as early peaks, so the network has not yet learned. After these 35 epochs, the network tries to balance the effectiveness with a constant increase in steps of 20 epochs.

In the training phase, the weights have not changed if the classification was correct. The iris dataset was divided as follows: 70% for training and 30% for testing. We think that the times when there are differences between training and testing are because the size of the dataset for testing was small for the 6260 network connections.

IX. Discussion

We present a discussion focused on the implementation, advantages of the proposed approach, and difficulties found in this project.

A. Implementation

In this work, we present two main points. The former is the network that can reproduce an output in order to stimulate muscles for the C. Elegans movements. Then, the presented approach can be used to simulate the movement of muscles (right and left) using a binary output. The latter is the memory that can be tested in a dynamic way in order to run the network with several scenarios to classify different patterns and guarantee that the network saved all patterns.

B. Advantages

Using this network we consider the following advantages:

- Using a neuronal network like a directed graph enables the algorithms and operations of the directed graph.
- The neuron connections are based on a real model. Thus, the signal propagation can use a branching factor

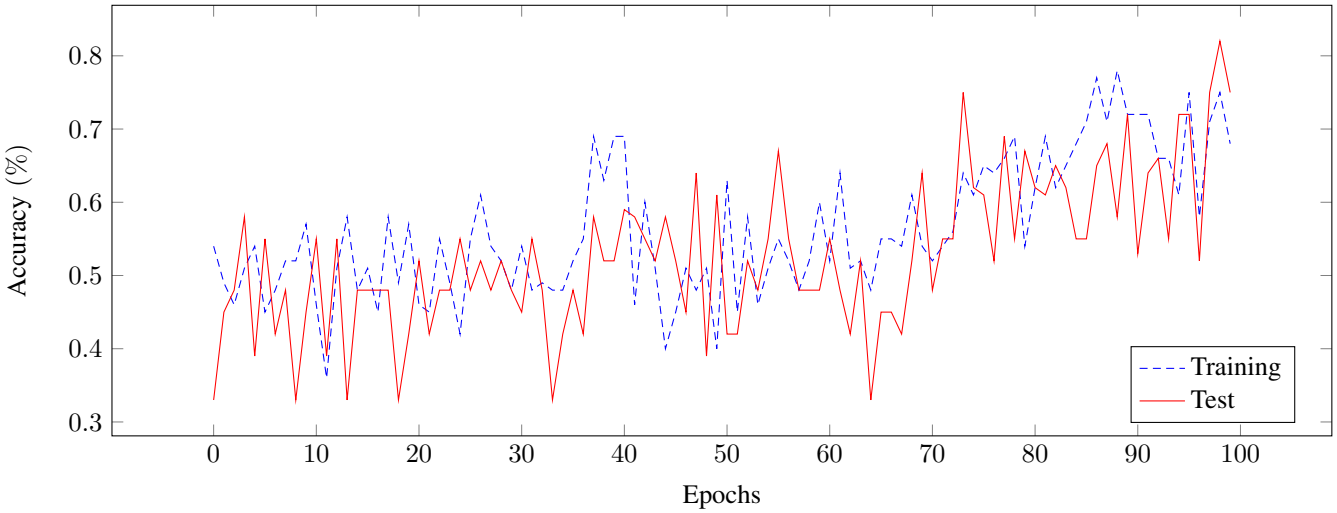

Fig. 9: **Accuracy results of training and testing the network:** These results represent the precision per epoch of training the neural network using the iris dataset. We used 100 epochs and for each epoch, the dimensions (150 rows x 3 attributes) in the dataset were the same, but we shuffled the data.

with spikes interaction in comparison to a network by layers.

- The number of neurons is constant to C. Elegans. Then, it is not necessary to create a random number of neurons.

C. Difficulties

The main difficulty of this work was using the HH model because this model does not have a good performance. The model takes between 20 and 140 seconds to calculate a membrane potential for a neuron. Thus, the training step was a challenge because it takes a long time to set the HH model for 6004 connections, the branching factor, and epochs.

The second difficulty of this work was generating the voltage peaks using the branching factor since the proposed algorithm uses recursion to generate the path propagation.

Another difficulty was finding the neural connections of C. Elegans since in some cases these connections were not complete. The analysis of the connections was proposed in a previous work [9], but modeling it using a directed graph was a big challenge.

Another difficulty was mixing the HH model and STDP in order to make the classification of patterns, because it is a novel method that calculates the membrane potential for each neuron, and then calculates the firing time in order to calculate the weight of connections in the training phase. Also, the way to use the weights and the spikes to predict the class for certain entries.

X. Conclusions and Future Work

The connections of the C. Elegans neural network allowed us to propose an experimental approach to train the network. In this way, we use the network to classify patterns using the STDP model. We tested with an iris dataset using the HH model with a time series (15 seconds in order to calculate the membrane potential and check the spikes). The network graph allows observing the neurons and their links. As a result, the network accuracy in 100 epochs was greater than 78% for training and testing for the iris dataset; then, we think that the result is related to the configuration of the HH model because the network chooses output neurons for each target class, and then for each output neuron the network calculates the spike activation to establish if the neuron predicts in a certain class. The average voltage was varied because of the potential membrane value for each neuron. For each experiment, the time was constant perhaps because the potential membrane depends on the spatio-temporal dataset encoding.

As future work, it is intended to implement the C. Elegans network and a classic artificial neural network for an accurate comparison, so that the networks are capable of classifying different encryption techniques. Initially, the networks will be trained with the techniques: substitution, affine, and vigenere. The inputs for the classification of the techniques are previously defined in the network with the purpose of performing a process that allows us to obtain an expected output, which in this case would be that the neural networks are able to identify the encryption technique that has been used. Such information will be provided to both networks with encrypted texts.

In addition, in future work, we will analyze both neural networks using different metrics like speed, efficiency, and accuracy to characterize the two networks. This will be done in order to know which network is better for each metric. For this purpose, we will propose several experiments to compare both networks. In this way, we intend to use both networks to classify cipher techniques such as Vigenere, DES, among others, based on an encrypted text.

References

[1] H. Lu, Y. Li, M. Chen, H. Kim, and S. Serikawa, "Brain intelligence: go beyond artificial intelligence," *Mobile Networks and Applications*, vol. 23, no. 2, pp. 368–375, 2018.

[2] S.-C. Wang, "Artificial neural network," in *Interdisciplinary Computing in Java Programming*. Springer, 2003, pp. 81–100.

[3] M. Hu and L. Wang, "Almost periodicity in shifts delta (+/-) on time scales and its application to hopfield neural networks." *Engineering Letters*, vol. 29, no. 3, pp. 864–872, 2021.

[4] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in Neuroscience*, vol. 12, p. 774, 2018.
[5] J. G. White, E. Southgate, J. N. Thomson, S. Brenner *et al.*, "The structure of the nervous system of the nematode caenorhabditis elegans," *Philos Trans R Soc Lond B Biol Sci*, vol. 314, no. 1165, pp. 1–340, 1986.
[6] J. Kunert, E. Shlizerman, and J. N. Kutz, "Low-dimensional functionality of complex network dynamics: Neurosensory integration in the caenorhabditis elegans connectome," *Physical Review E*, vol. 89, no. 5, p. 052805, 2014.
[7] A. Azulay, E. Itskovits, and A. Zaslaver, "The c. elegans connectome consists of homogenous circuits with defined functional roles," *PLoS Computational Biology*, vol. 12, no. 9, p. e1005021, 2016.
[8] J. Kim, W. Leahy, and E. Shlizerman, "Neural interactome: Interactive simulation of a neuronal system," *Frontiers in Computational Neuroscience*, vol. 13, p. 8, 2019.
[9] J. Hernandez and H. Florez, "An experimental comparison of algorithms for nodes clustering in a neural network of caenorhabditis elegans," in *International Conference on Computational Science and Its Applications*. Springer, 2021, pp. 327–339.
[10] K. Kumarasinghe, N. Kasabov, and D. Taylor, "Brain-inspired spiking neural networks for decoding and understanding muscle activity and kinematics from electroencephalography signals during hand movements," *Scientific Reports*, vol. 11, no. 1, pp. 1–15, 2021.
[11] S. Ghoshdastidar and H. Adeli, "Spiking neural networks," *International Journal of Neural Systems*, vol. 19, no. 04, pp. 295–308, 2009.
[12] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.
[13] R. Appali, U. Van Rienen, and T. Heimburg, "A comparison of the hodgkin–huxley model and the soliton theory for the action potential in nerves," in *Advances in Planar Lipid Bilayers and Liposomes*. Elsevier, 2012, vol. 16, pp. 275–299.
[14] M. Begum, S. B. Hafiz, J. Islam, and M. J. Hossain, "Long-term software fault prediction with robust prediction interval analysis via refined artificial neural network (rann) approach." *Engineering Letters*, vol. 29, no. 3, pp. 1158–1171, 2021.
[15] M. A. Bhuiyan, R. Jalasutram, and T. M. Taha, "Character recognition with two spiking neural network models on multicore architectures," in *2009 IEEE Symposium on Computational Intelligence for Multimedia Signal and Vision Processing*. IEEE, 2009, pp. 29–34.
[16] W. Gerstner, R. Kempter, J. L. Van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 6595, pp. 76–78, 1996.
[17] J. Wang, A. Belatreche, L. Maguire, and T. M. McGinnity, "An online supervised learning method for spiking neural networks with adaptive structure," *Neurocomputing*, vol. 144, pp. 526–536, 2014.
[18] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains," *PloS One*, vol. 3, no. 1, p. e1377, 2008.
[19] A. Sboev, D. Vlasov, R. Rybka, and A. Serenko, "Spiking neural network reinforcement learning method based on temporal coding and stdp," *Procedia Computer Science*, vol. 145, pp. 458–463, 2018.
[20] J. Albers and A. Offenhäusser, "Signal propagation between neuronal populations controlled by micropatterning," *Frontiers in Bioengineering and Biotechnology*, vol. 4, p. 46, 2016.
[21] L. Galluccio, S. Palazzo, and G. E. Santagati, "Characterization of signal propagation in neuronal systems for nanomachine-to-neurons communications," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2011, pp. 437–442.
[22] C. A. Moreira and M. A. de Aguiar, "Modular structure in c. elegans neural network and its response to external localized stimuli," *Physica A: Statistical Mechanics and its Applications*, vol. 533, p. 122051, 2019.
[23] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Computation*, vol. 19, no. 11, pp. 2881–2912, 2007.
[24] S. R. Wicks, C. J. Roehrig, and C. H. Rankin, "A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria," *Journal of Neuroscience*, vol. 16, no. 12, pp. 4017–4031, 1996.
[25] G. Wang, Y. Zeng, and B. Xu, "A spiking neural network based autonomous reinforcement learning model and its application in decision making," in *International Conference on Brain Inspired Cognitive Systems*. Springer, 2016, pp. 125–137.
[26] D. Sanchez and H. Florez, "Improving game modeling for the quoridor game state using graph databases," in *International Conference on Information Technology & Systems*. Springer, 2018, pp. 333–342.
[27] Kaggle, "Iris flower dataset," https://www.kaggle.com/datasets/arshid/iris-flower-dataset, 2018.
[28] G. Yan, P. E. Vértes, E. K. Towlson, Y. L. Chew, D. S. Walker, W. R. Schafer, and A.-L. Barabási, "Network control principles predict neuron function in the caenorhabditis elegans connectome," *Nature*, vol. 550, no. 7677, pp. 519–523, 2017.
[29] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii, "Structural properties of the caenorhabditis elegans neuronal network," *PLoS Computational Biology*, vol. 7, no. 2, p. e1001066, 2011.

Jorge Hernandez is postgraduate student in information and communication sciences at the Universidad Distrital Francisco Jose de Caldas. This article presents some of the results of his master's research project.

Karen Daza is undergraduate student in telematics engineering at the Universidad Distrital Francisco Jose de Caldas. She contributed to the experiments presented in the article.

Hector Florez is Full Professor at the Universidad Distrital Francisco Jose de Caldas, Bogota, Colombia. He is Ph.D. in Engineering at the Universidad de Los Andes. His research interests are model-driven engineering, artificial intelligence, and data analytics.