Neuroidal Network Software Development Class Project for Detection of Arrhythmia in Patients Experiencing Heart Failure

CSC 9010, Summer 2021
July 29, 2021

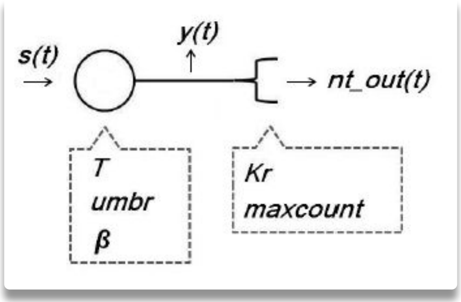
Overview

- What is a Neuroid
    - Schematic and Parametric Representation
- User Flow Design
- Simulator Prototype and User Flow Diagram
- Front-End Development
- Collaboration with UI Team
- Back-End Development
- Software Requirements

What is a Neuroid?

- Computational component to mimic biological neurons
- Improvement over **perceptron** of a neural network
- 3 phases:
    - Comparator
    - Pulse Modulation
    - Pulse Demodulation

Schematic and Parametric Representation of a Neuroid

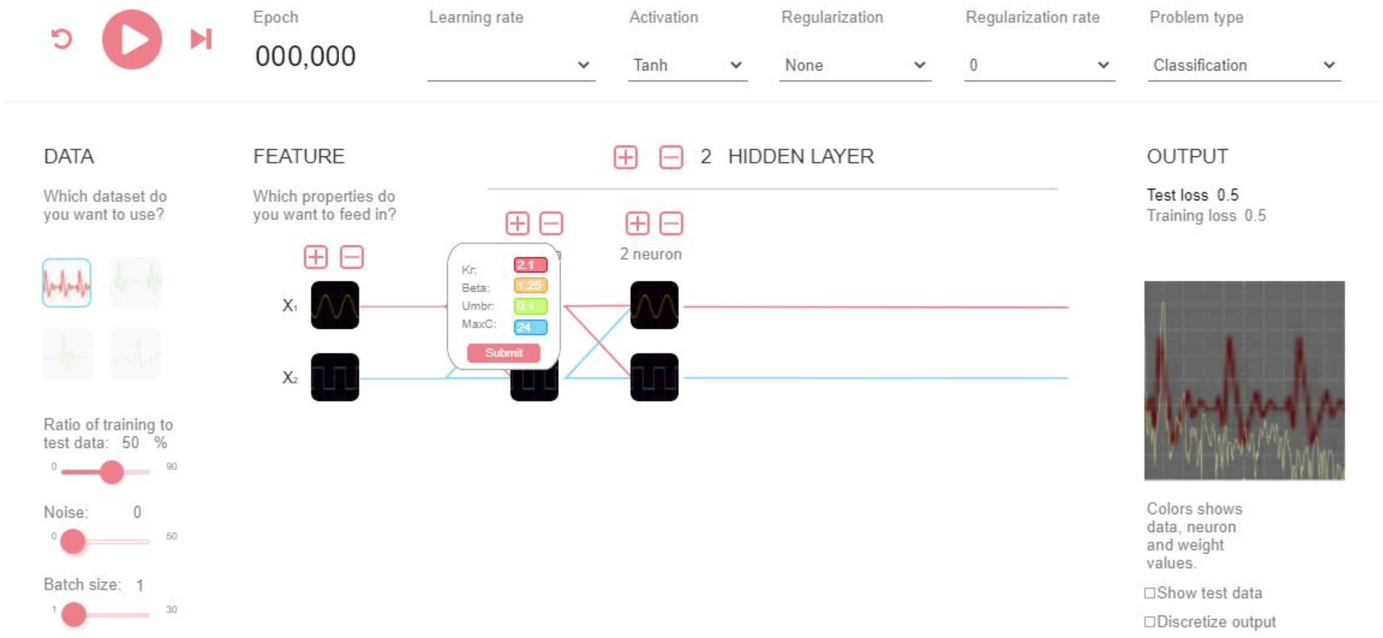


s(t)	Depolarization reaching trigger zone
y(t)	Output of the pulse frequency modulator block
nt_out(t)	Output signal
T	Time interval between two consecutive pulses
umbr	Minimum Threshold
β	Proportionality Constant
Kr	Regeneration Constant
maxcount	A halting value to stop the indefinite extension of the output signal

User Flow Design Phase

- Develop a prototype/digital representation of the simulator layout
- Software Used: AXURE RP
- AXURE RP:
    - Software for creating prototypes and specifications for websites/applications using drag-drop features, formatting of widgets.
    - Flexible in modifying and altering designs
    - Highly interactive
    - Cross-platform

Simulator Prototype



Epoch 000,000

Learning rate

Activation: Tanh

Regularization: None

Regularization rate: 0

Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50 %

Noise: 0

Batch size: 1

FEATURE

Which properties do you want to feed in?

2 HIDDEN LAYER

2 neuron

Kr: 2.1

Beta: 1.25

Umbr: 0.1

MaxC: 24

OUTPUT

Test loss 0.5

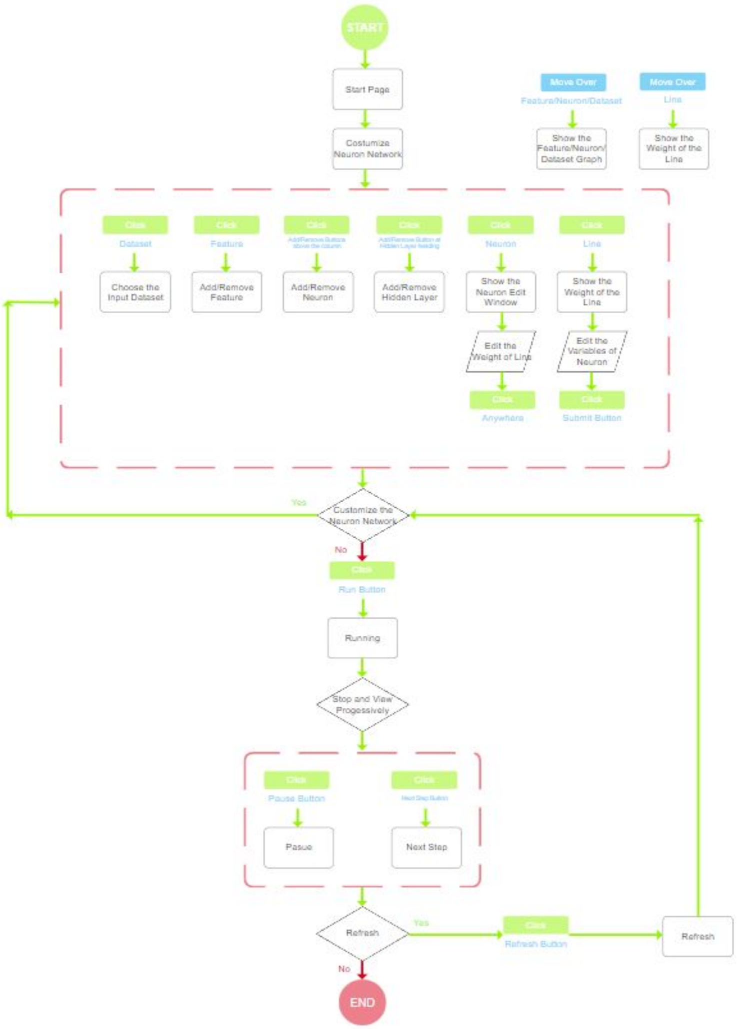
Training loss 0.5

Colors shows data, neuron and weight values.

□ Show test data

□ Discretize output

User Flow Diagram

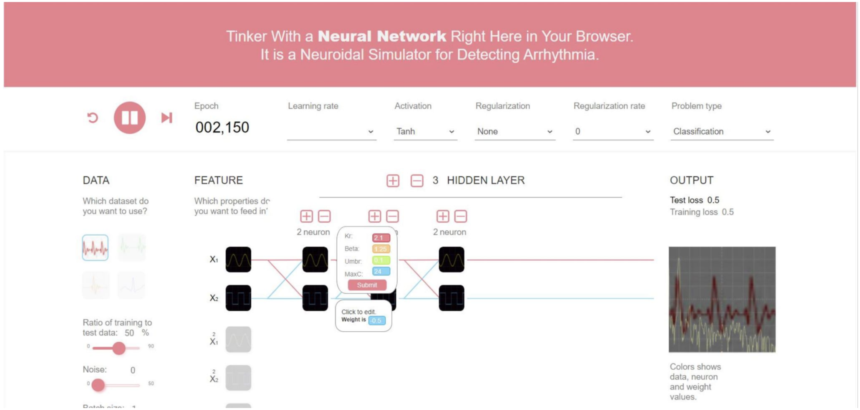


Front-End Development Goals

- Assess business requirements
- Coordinate the UI design with the user flow team
- Coordinate the implementation with the back-end team
- Choose a tech stack
- Create options to quickly and easily change the inputs of the simulator
- Map out the neural network on the page
- Allow the user to have dynamic control over the simulation
- Display the output of the simulation alongside the network that updates as the simulation runs

Collaboration with the UI Team

- Frequent communication with the UI team
- Demo the designs regularly, walk them through the new developments
- Asked for feedback and kept an open mind for suggestions



Tinker With a **Neural Network** Right Here in Your Browser.
It is a Neuroidal Simulator for Detecting Arrhythmia.

Epoch 002,150

Learning rate

Activation: Tanh

Regularization: None

Regularization rate: 0

Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50 %

Noise: 0

FEATURE

Which properties do you want to feed in?

3 HIDDEN LAYER

Kr: 2.1

Beta: 1.25

Umbr: 0.1

MaxC: 24

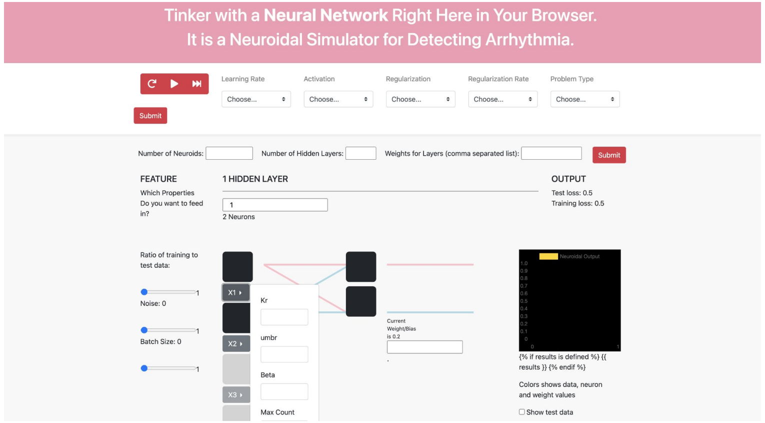
Click to edit. **Weight is** -0.5

OUTPUT

Test loss 0.5

Training loss 0.5

Colors shows data, neuron and weight values.



Tinker with a **Neural Network** Right Here in Your Browser.
It is a Neuroidal Simulator for Detecting Arrhythmia.

Learning Rate: Choose...

Activation: Choose...

Regularization: Choose...

Regularization Rate: Choose...

Problem Type: Choose...

Number of Neuroids:

Number of Hidden Layers:

Weights for Layers (comma separated list):

FEATURE

Which Properties Do you want to feed in?

Ratio of training to test data:

Noise: 0

Batch Size: 0

1 HIDDEN LAYER

1

2 Neurons

Kr

umbr

Beta

Max Count

Current Weight/Bias is 0.2

OUTPUT

Test loss: 0.5

Training loss: 0.5

{% if results is defined %} {{ results }} {% endif %}

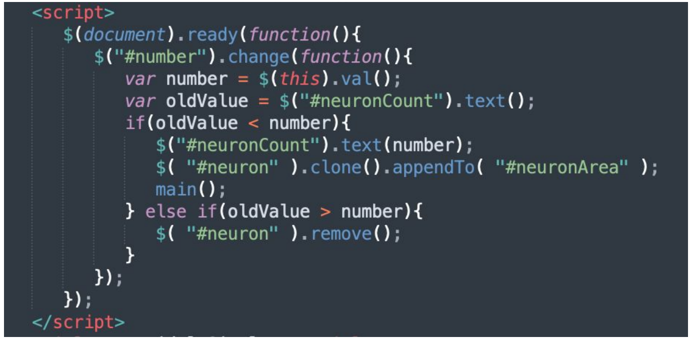
Colors shows data, neuron and weight values

☐ Show test data

Software Tools Utilized

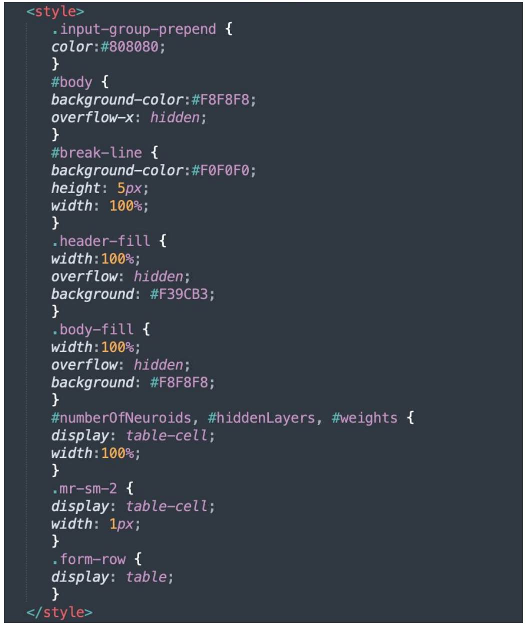
- HTML/CSS
- Bootstrap
- JavaScript
- Python
- Flask
- jQuery
- GitHub

Custom jQuery Function



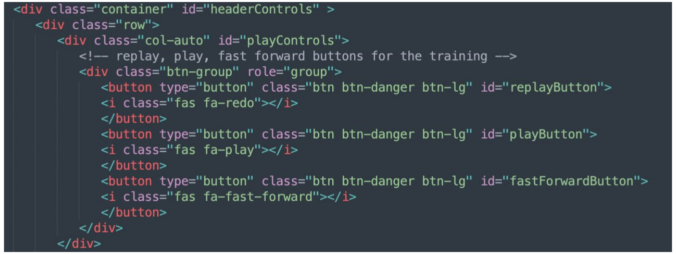
```
<script>
    $(document).ready(function(){
        $("#number").change(function(){
            var number = $(this).val();
            var oldValue = $("#neuronCount").text();
            if(oldValue < number){
                $("#neuronCount").text(number);
                $( "#neuron" ).clone().appendTo( "#neuronArea" );
                main();
            } else if(oldValue > number){
                $( "#neuron" ).remove();
            }
        });
    });
</script>
```

CSS



```
<style>
    .input-group-prepend {
    color:#808080;
    }
    #body {
    background-color:#F8F8F8;
    overflow-x: hidden;
    }
    #break-line {
    background-color:#F0F0F0;
    height: 5px;
    width: 100%;
    }
    .header-fill {
    width:100%;
    overflow: hidden;
    background: #F39CB3;
    }
    .body-fill {
    width:100%;
    overflow: hidden;
    background: #F8F8F8;
    }
    #numberOfNeuroids, #hiddenLayers, #weights {
    display: table-cell;
    width:100%;
    }
    .mr-sm-2 {
    display: table-cell;
    width: 1px;
    }
    .form-row {
    display: table;
    }
</style>
```

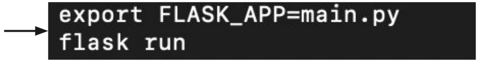
Bootstrap



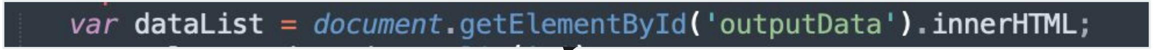
```
<div class="container" id="headerControls" >
    <div class="row">
        <div class="col-auto" id="playControls">
            <!-- replay, play, fast forward buttons for the training -->
            <div class="btn-group" role="group">
                <button type="button" class="btn btn-danger btn-lg" id="replayButton">
                <i class="fas fa-redo"></i>
                </button>
                <button type="button" class="btn btn-danger btn-lg" id="playButton">
                <i class="fas fa-play"></i>
                </button>
                <button type="button" class="btn btn-danger btn-lg" id="fastForwardButton">
                <i class="fas fa-fast-forward"></i>
                </button>
            </div>
        </div>
```

Collaboration with Backend Team

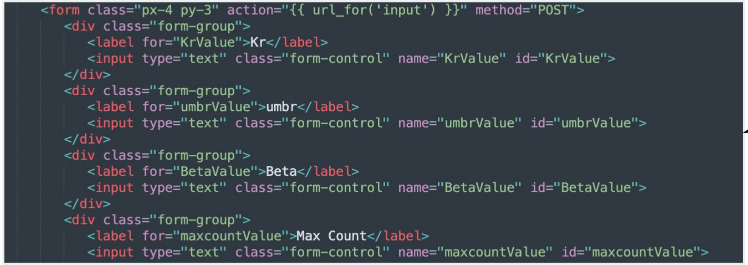
Terminal command to run program



```
export FLASK_APP=main.py
flask run
```

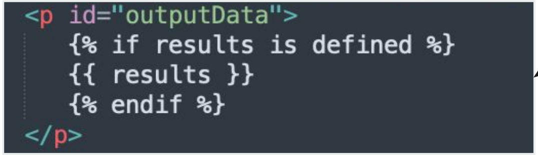


```
var dataList = document.getElementById('outputData').innerHTML;
```

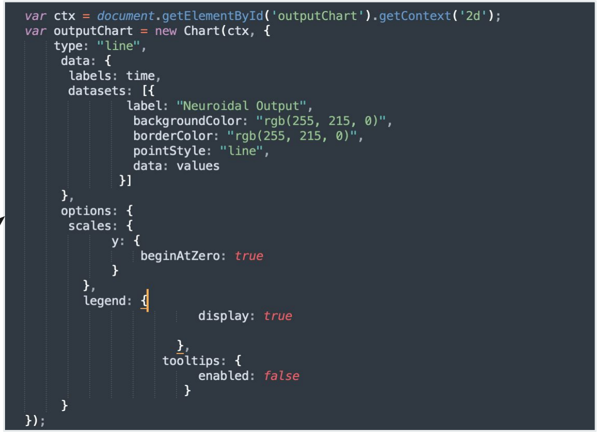


```
<form class="px-4 py-3" action="{{ url_for('input') }}" method="POST">
    <div class="form-group">
        <label for="KrValue">Kr</label>
        <input type="text" class="form-control" name="KrValue" id="KrValue">
    </div>
    <div class="form-group">
        <label for="umbrValue">umbr</label>
        <input type="text" class="form-control" name="umbrValue" id="umbrValue">
    </div>
    <div class="form-group">
        <label for="BetaValue">Beta</label>
        <input type="text" class="form-control" name="BetaValue" id="BetaValue">
    </div>
    <div class="form-group">
        <label for="maxcountValue">Max Count</label>
        <input type="text" class="form-control" name="maxcountValue" id="maxcountValue">
```

Getting output from backend to the frontend



```
<p id="outputData">
    {% if results is defined %}
    {{ results }}
    {% endif %}
</p>
```



```
var ctx = document.getElementById('outputChart').getContext('2d');
var outputChart = new Chart(ctx, {
    type: "line",
     data: {
      labels: time,
      datasets: [{
              label: "Neuroidal Output",
               backgroundColor: "rgb(255, 215, 0)",
               borderColor: "rgb(255, 215, 0)",
               pointStyle: "line",
               data: values
             }]
     },
     options: {
      scales: {
            y: {
                beginAtZero: true
            }
        },
        legend: {
                        display: true

                     },
                   tooltips: {
                        enabled: false
                      }
     }
});
```

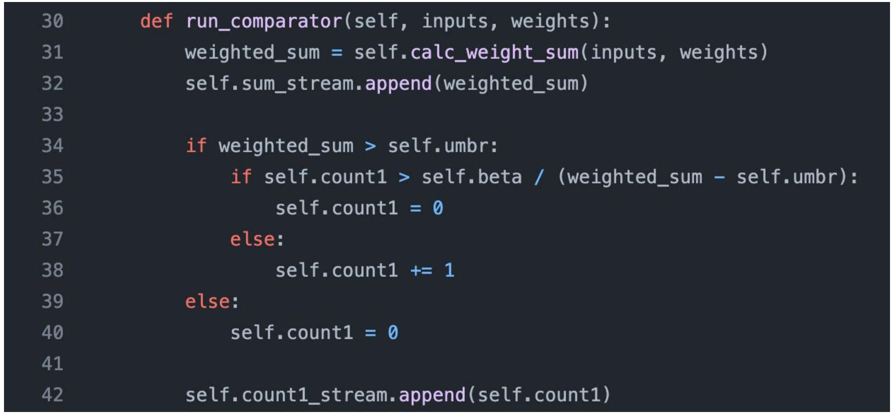
Using Chart.js to create output graph

Backend Development Goals

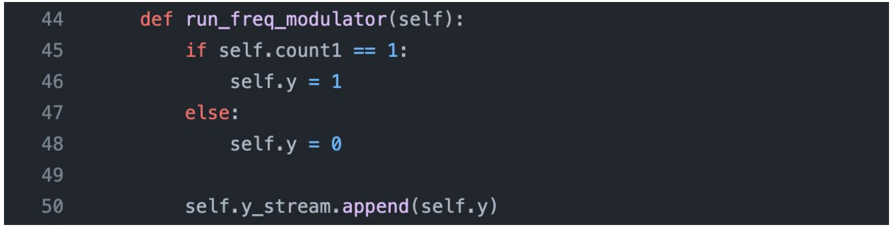
- Generate results similar to the neuroid presented in Erick’s paper
- Verify results and ensure program has minimal bugs
- Identify areas of improvement in terms of the algorithm
- Integrate seamlessly with the front-end

Neuroid Logic

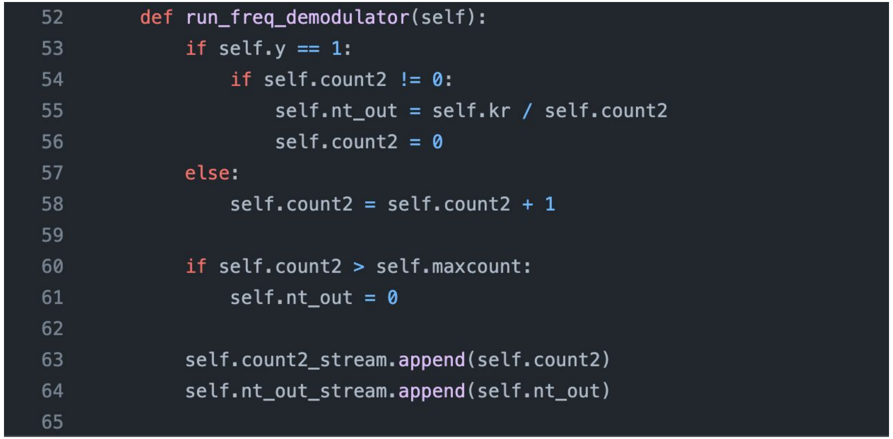
- Comparator
- Frequency Modulator
- Frequency Demodulator



```python
    def run_comparator(self, inputs, weights):
        weighted_sum = self.calc_weight_sum(inputs, weights)
        self.sum_stream.append(weighted_sum)

        if weighted_sum > self.umbr:
            if self.count1 > self.beta / (weighted_sum - self.umbr):
                self.count1 = 0
            else:
                self.count1 += 1
        else:
            self.count1 = 0

        self.count1_stream.append(self.count1)
```



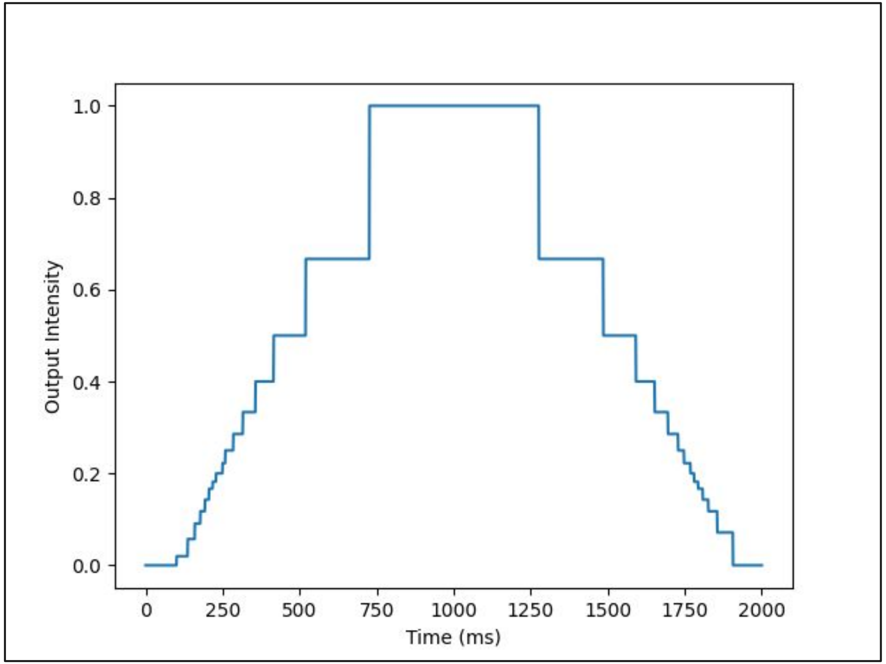
```python
    def run_freq_modulator(self):
        if self.count1 == 1:
            self.y = 1
        else:
            self.y = 0

        self.y_stream.append(self.y)
```



```python
    def run_freq_demodulator(self):
        if self.y == 1:
            if self.count2 != 0:
                self.nt_out = self.kr / self.count2
                self.count2 = 0
        else:
            self.count2 = self.count2 + 1

        if self.count2 > self.maxcount:
            self.nt_out = 0

        self.count2_stream.append(self.count2)
        self.nt_out_stream.append(self.nt_out)

```

Neuroid Results

- umbr = 0.1
- beta = 1.25
- kr = 2
- maxcount = 50
- t = 1



Requirements

System Feature 1: User Flow

Description

- The user interacts with the software by entering values in editable text fields or selecting from drop down menus.
- The user can view processing time of
- The user can view outputs generated by the software in graphical format.

Requirements

- The neuroidal network must implement hidden neuroid layers.
- Time parameter must be visible and run real time when the execute button is started.
- Question text boxes must be visible and be present to guide user on how to interact with form and features.r

Requirements (cont.)

System Feature 2: Front-End

Description

- The inputs that are provided by the user associates with values connected to the back-end for processing.
- Form functionality is available allowing the user to interact with the software.

Requirements

- “Execute” button must run back-end code to generate output
- “Refresh” button must reset front-end form and clear stored values from previous trial
- “Forward” or “Next” button must cycle through code for next iteration in signal processing
- Number of neuroids that can be entered must be a positive integer
- Headers must match variable association in back-end code
    - User must be able to select from list of inputs or have field available for user defined input
- “Test Loss” and “Training Loss” must be present to user after trial is executed
- Training properties must be able to be modified before trial is executed using slider object
    - Training properties must include “Ratio of Training to Test Data”, “Noise”, and “Batch Size”

Requirements (cont.)

System Feature 3: Connectivity

Description

- The front-end user interface button and fields must connect to back-end code, so software may function as intended.

Requirements

- Inputs entered by the user must transfer to back-end code for algorithm processing.
- Outputs generated by back-end code must transfer to the front-end form output section for graphical presentation.
- Graphical presentation must exhibit input signal and output signal.
- Form values must match neuroid parameters.

Requirements (cont.)

System Feature 4: Back-End and Neuroid

Description

- The neuroid is the most basic functioning unit of the neuroidal network.
- The neuroid can perform three operations that are carried out to process incoming information such as comparison, frequency pulse modulation, and frequency pulse demodulation.
- Back-end team implements the processing of the neuroid that extracts form inputs and processes them using signal inputs defined by the user.

Requirements

- The neuroid must have three main functions: a comparator, a frequency modulator, and a frequency demodulator.
    - The comparator must compare the incoming signal against the activation threshold.
    - The incoming signal must be represented by any value between -1 and 1.
    - Frequency modulator must facilitate downstream neuroid.
        - The frequency modulator must generate an impulse train with a frequency that varies proportionally to the input amplitude.
    - Frequency demodulator must inhibit downstream neuroid.
        - The frequency demodulator must be able to demodulate the impulse train.

Requirements (cont.) - Feature 4

- The neuroid must be able to represent normally silent and tonically active neurons.
- The neuroid must provide an amplitude-discrete version of the input signal with variable adjustment
- A threshold value must be established for the neuroid
  - If the threshold is surpassed, the frequency modulator or demodulator must be activated.
  - If the threshold is not surpassed, the outcome must be zero.
- Back-end development team implements the processing of the neuroid that extracts inputs.
- “Test Loss” and “Training Loss” must be generated for variable output to user on front-end