NERVOUS Systems Tutorial - Artificial Neural Microcircuits

This part of the tutorial concerns NERVOUS' first experiments with the artificial neural microcircuit concept. Using the Python script provided, you will be able to observe the operation of a few of the microcircuits thus far produced.

To follow this tutorial document, you will need the following:

1. An installation of Python 3.10, along with the following packages:
    a. Brian2 version 2.5.1
    b. numpy version 1.23.3
    c. matplotlib version 3.6.1

2. The "NERVOUS Systems Tutorial - Artificial Neural Microcircuits" .zip, containing:
    a. The Brian2_Wrapper.py Python script
    Brian2 is a Python & C++ based Spiking Neural Network (SNN) simulator. This script uses Brian2 to assemble a microcircuit (according to the configuration in the file passed to it as a parameter), before then applying the test stimulus to it (stored in the stimulus_indices and stimulus_times files). The spike trains corresponding to the inputs to and outputs of the microcircuit will then be displayed as plots.

    b. A simple 2x2 grid network config file (named 2x2_Grid.txt)
    These files contain the configuration for a 2x2 grid of spiking neurons, where the two neurons on one side are inputs and a fifth neuron attached to the other side is the output.

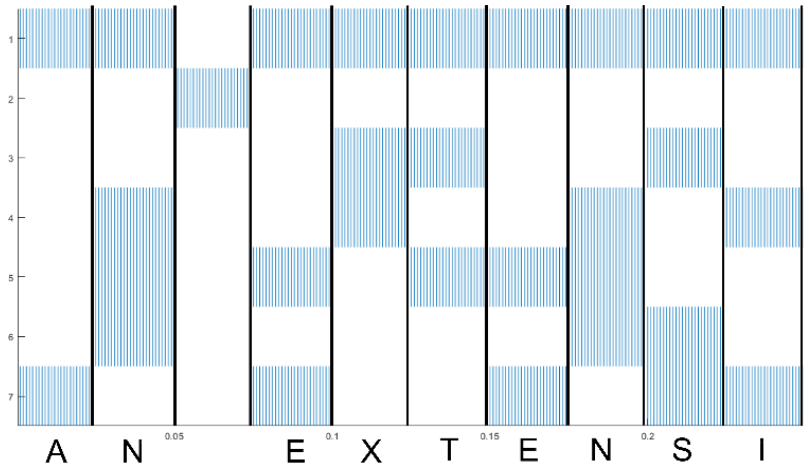
    c. A set of microcircuit config files (named in the format Microcircuit_[number].txt)
    These files contain the configurations for the various microcircuits that can be assembled and tested using the Python script.

    Both the grid network and microcircuit configuration files have the following format:
        1. Number of Inputs
        2. Number of Neurons
        3. Number of Outputs
        4. A number of lines containing the input weight matrix
        5. A number of lines containing the internal connection matrix
        6. A number of lines containing the output connections

d. .png images of the structures of the network & microcircuits
   These files provide illustrations of the network & various microcircuits described by the config .txt files, and are named accordingly.
e. The stimulus_indices.txt and stimulus_times.txt files
   These files contain the test stimulus that will be applied to the network or microcircuit. The times file contains the times at which input spikes occur, while the indices file contains the ids of the input those spikes occur at. For example, an index entry of 1, 7, 2, 4 and a times entry of 0.5, 0.5, 0.6, 0.7 would correspond to a spike at inputs 1 & 7 at time 0.5, followed by a spike at input 2 at time 0.6, and finally a spike at input 4 at time 0.7.

   The stimulus for the 2x2 grid network, in the files prefixed “2_bit_”, consists of a pair of spike trains 500 ms long. One spike train has a fixed frequency, while the second starts the same as the first, but slowly diverges.

   The stimulus for the microcircuits, in the files prefixed “8_bit_”, represent a stream of text, translated into UTF-8 values and then encoded as bursts of spikes. Each burst lasts 25ms, resulting in a full duration of 50750ms. The full text can be found in appendix 1 of this document.



Execution

To run the Python script enter the following command:

python3 Brian2_Wrapper.py [**file_name**] [**-d stimulus_duration**] [**-p plot_mode**]

After a short period of time, a window will appear, displaying the input and output spike trains produced by the execution of the microcircuit.

- file_name is the name of the configuration file for the network you want to use (including the .txt extension);
- -d stimulus_duration is an optional parameter to specify how long you want to run the network and apply stimulus to it. If not given a stimulus_duration, or a duration is provided that exceeds the duration of the stimulus, the script will run for the full duration of the test stimulus;
- -p plot_mode is an optional parameter to specify what spike trains you want to see when the script finishes. If left blank or set to "simple", only plots of the input & output spike trains will be displayed; if set as "full", an additional window will display the spike trains of all the neurons in the network or microcircuit.

If you repeat this process with a few different configuration files, you should see how differently structured spiking networks respond to the stimulus.

Microcircuits

Each of the microcircuits respond to one or more patterns of input spike trains, which of course correspond to different characters in the text stream. While this might be hard to see from the output full plots of all the spike trains, the match ups are as follows:

- Microcircuit 49: Full Stops
- Microcircuit 67: Comas, Dashes & Full Stops
- Microcircuit 392: Dashes
- Microcircuit 466: Spaces, Comas, Dashes & Full Stops

Appendix 1 - Text Stream

AN EXTENSION OF ARTIFICIAL GENE REGULATORY NETWORKS ARTIFICIAL EPIGENETIC NETWORKS IMPLEMENT AN ADDITIONAL LAYER OF BIO-INSPIRED CONTROL TO ALLOW FOR ENHANCED PERFORMANCE ON CERTAIN TYPES OF CONTROL TASKS BY FACILITATING TOPOLOGICAL SELF-MODIFICATION. THIS WORK LOOKS TO EXPAND THE APPLICATIONS OF ARTIFICIAL EPIGENETIC NETWORKS BY TRANSLATING THE EXISTENT SOFTWARE ARCHITECTURE INTO A FORM SUITABLE FOR IMPLEMENTATION ON A FIELD PROGRAMMABLE GATE ARRAY. THIS OPENS THE POSSIBILITY OF ARTIFICIAL EPIGENETIC NETWORKS BEING USED IN APPLICATIONS WHERE HIGH-PERFORMANCE COMPUTATIONAL RESOURCES ARE IMPRACTICAL, SUCH AS ROBOTIC CONTROL. THIS THESIS DEVELOPS A MORE RESOURCE EFFICIENT ARCHITECTURE FOR EPIGENETIC NETWORKS BASED ON REDUCED PRECISION INTEGER MATHEMATICS, AND THEN TRANSLATES IT INTO HARDWARE TO PROVIDE IMPROVEMENTS IN RESOURCE UTILISATION AND EXECUTION SPEED WHILE NOT SACRIFICING THE UNIQUE BENEFITS PROVIDED BY THE EPIGENETIC MECHANISMS. THE APPLICATION TO ROBOTIC CONTROL IS INVESTIGATED BY UTILISING THE HARDWARE ARTIFICIAL EPIGENETIC NETWORK TO PERFORM VARIOUS VERSIONS OF A FORAGING TASK, CULMINATING IN ONE DESIGNED TO REPLICATE A SEARCH AND RESCUE SCENARIO. WHILE THE ARTIFICIAL EPIGENETIC NETWORKS DID NOT DEMONSTRATE SIGNIFICANT PERFORMANCE IMPROVEMENTS COMPARED TO THEIR NON-EPIGENETIC COUNTERPARTS, THIS DID INDICATE THAT NOT EVERY TYPE OF CONTROL TASK BENEFITS FROM THE INCLUSION OF THE EPIGENETIC MECHANISM. IN ADDITION, THIS WORK INVESTIGATES ANOTHER ASPECT OF ARTIFICIAL EPIGENETIC NETWORKS, SPECIFICALLY THE LIMITS OF THEIR TOPOLOGICAL SELF-MODIFICATION WITH RESPECT TO REACTING TO CHANGES IN THEIR ENVIRONMENT. MORE SPECIFICALLY, IT IS ASKED IF AN ARTIFICIAL EPIGENETIC NETWORK CAN MAINTAIN ITS ABILITY TO PERFORM A SPECIFIC TASK WHEN CONFRONTED WITH FACTORS OUTSIDE OF THOSE IT HAS BEEN OPTIMISED TO HANDLE. WHILE NOT CONCLUSIVELY DEMONSTRATED, THERE IS SUFFICIENT EVIDENCE THAT THE ANSWER TO THIS QUESTION DEPENDS ON THE PERFORMANCE GAINS IMPARTED BY EPIGENETIC BEHAVIOURS UNDER NORMAL CIRCUMSTANCES.