Name: Pooria Heydarian
Student.no: 610399206

Computational Neuroscience

Project three report

Understanding Information Encoding and Learning Processes in Neuroscience

1. Implementation of Different Modes of Encoding

1.1 Time to First Spike Encoding:

Description: Time to first spike encoding is a method where information is encoded based on the time it takes for a neuron to generate its first action potential (spike) in response to a stimulus. This encoding scheme is particularly relevant in scenarios where the timing of neural responses carries significant information.

Implementation Details: Time to First Spike (TTFS) encoding was implemented using a Python function ttfs_encode. This function takes as input a numpy array of numerical data and a threshold value, representing the threshold for spiking a neuron. The function returns a numpy array of spike times for each data point.

Algorithm: The implementation iterates through each data point and simulates the increase in membrane potential over time. If the membrane potential reaches or exceeds the specified threshold, the spike time for the neuron is recorded. The process continues until the spike time is determined for each neuron.

Example Usage and Visualization: To demonstrate the TTFS encoding, random data between 0 and 1 was generated as input. The generated spike times were then visualized using a raster plot, which depicts the spike times of neurons over time. Each spike is represented as a vertical line in the raster plot, indicating the timing of neural responses to the input stimuli.

Expected Outcome: The implementation of TTFS encoding is expected to provide insights into the temporal dynamics of neural responses to stimuli. By capturing the precise timing of spikes, this encoding scheme facilitates the representation of time-sensitive information in neural networks.

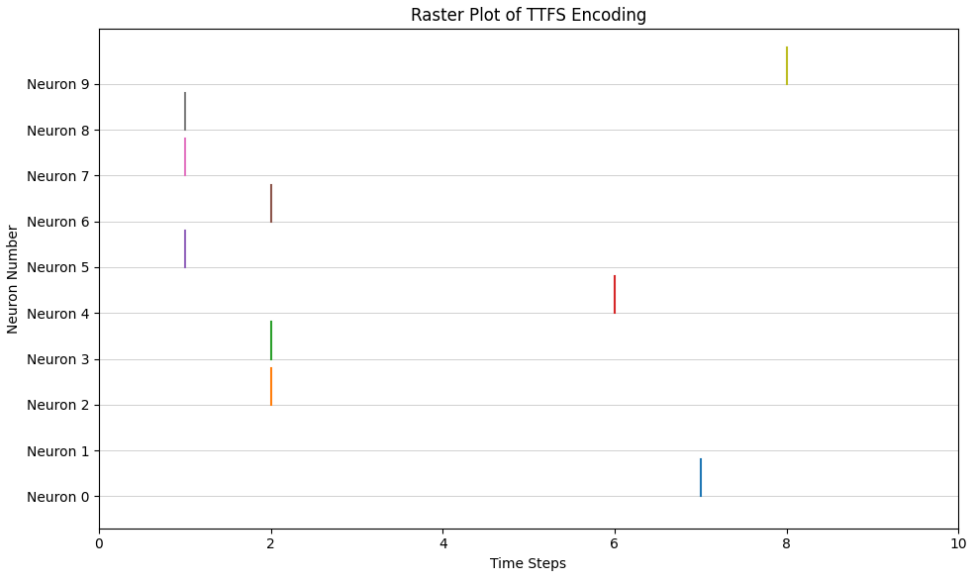


Figure 1 Rastor plot Time to First Spike

Name: Pooria Heydarian
Student.no: 610399206

Computational Neuroscience

Project three report

Effects of Time-to-First-Spike Encoding Parameter

1. Threshold Value:

- **Effect:** The threshold value determines the level of membrane potential required to trigger the first spike in a neuron.
- **Impact:**
    - Higher threshold values increase the required input intensity for spike generation, resulting in fewer spikes and potentially sparser representations of input data.
    - Lower threshold values lower the activation threshold, leading to more frequent spike generation and denser representations of input patterns.

2. Encoding Fidelity:

- Effect: Time-to-first-spike encoding translates input data into temporal patterns of spike activation, influenced by the threshold value.
- **Impact:**
    - Higher threshold values may result in more selective encoding, with only strong input signals eliciting spikes and contributing to neuronal activity.
    - Lower threshold values broaden the range of input intensities that trigger spikes, capturing a wider range of input features but potentially increasing overlap between encoded patterns.

3. Dynamic Range:

- **Effect:** The dynamic range of time-to-first-spike encoding refers to the range of input intensities that elicit distinct spike timings.
- **Impact:**
    - Narrow dynamic ranges, associated with higher threshold values, limit the sensitivity of encoding to subtle variations in input signals, emphasizing coarse discrimination between distinct input levels.
    - Wide dynamic ranges, achieved with lower threshold values, enhance the sensitivity of encoding, allowing for finer discrimination between input intensities and more nuanced representations of input features.

4. Spike Timing Precision:

- Effect: The precision of spike timing in time-to-first-spike encoding influences the temporal resolution of encoded information.
- **Impact:**
    - Higher threshold values result in fewer, but more precisely timed spikes, enhancing the temporal precision of encoded patterns and facilitating precise timing-based computations.
    - Lower threshold values may lead to more variable spike timings, reducing temporal precision but potentially enabling robust encoding of input dynamics and temporal patterns.

1.2 Rate Encoding

Description: Rate encoding is a method where information is encoded based on the firing rate of neurons over a certain period of time. In this encoding scheme, the frequency of neural spikes conveys information about the magnitude or intensity of the stimulus.

Implementation Details: The Rate Encoding was implemented to capture the frequency of neural spikes in response to stimuli. Unlike Time to First Spike encoding, which focuses on the timing of the first spike, Rate Encoding considers the overall firing rate of neurons over a specified time window.

Algorithm: The implementation of Rate Encoding involves computing the firing rate of neurons by counting the number of spikes within a given time window. This firing rate information is then used to encode the input stimuli, where higher firing rates represent higher stimulus intensities.

Example Usage and Visualization: Similar to the Time to First Spike encoding, random data between 0 and 1 was generated as input for Rate Encoding. The firing rates of neurons were computed over a specified time window, and the resulting encoding was visualized using appropriate plots or representations.

Expected Outcome: Rate Encoding is expected to provide a continuous representation of stimulus intensity by capturing the frequency of neural activity over time. This encoding scheme enables the discrimination of stimuli with varying magnitudes based on the firing rates of neurons.

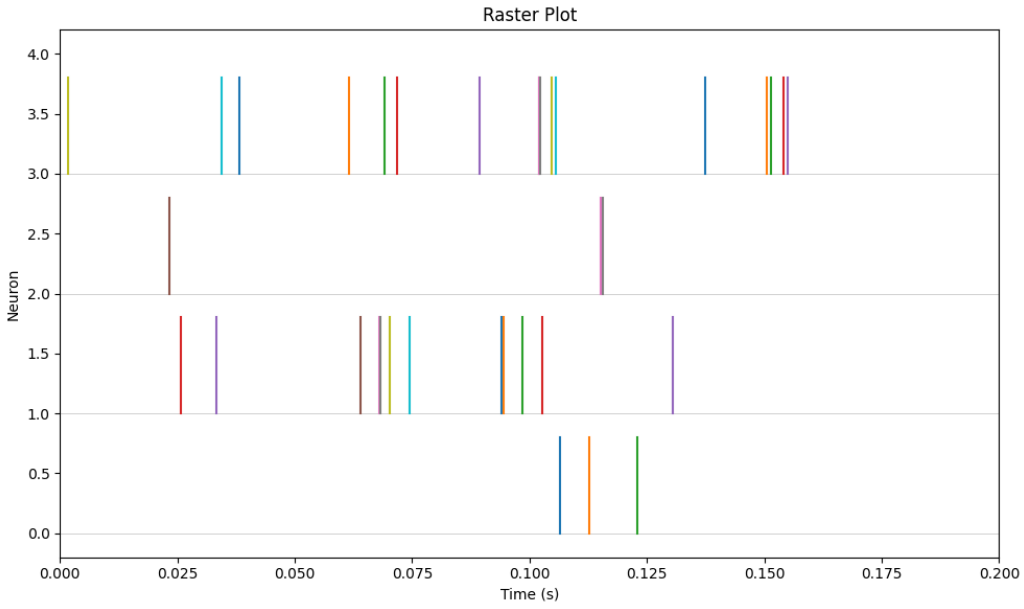


Figure 2 Raster plot Rate Encoding

Name: Pooria Heydarian
Student.no: 610399206

Computational Neuroscience

Project three report

Effects of Parameters in Rate Encoding

1. Minimum Firing Rate (rate_min):

- **Effect:** Rate_min determines the minimum firing rate of neurons encoding the lowest input values.
- **Impact:**
    - Higher values of rate_min lead to higher baseline activity levels, ensuring that even low input values elicit some neural response.
    - Lower values of rate_min result in lower baseline activity, requiring stronger input signals to elicit neuronal firing and encode information.

2. Maximum Firing Rate (rate_max):

- **Effect:** Rate_max sets the maximum firing rate of neurons encoding the highest input values.
- **Impact:**
    - Higher values of rate_max increase the dynamic range of firing rates, allowing neurons to encode a wider range of input intensities.
    - Lower values of rate_max compress the dynamic range, limiting the range of firing rates and potentially reducing the discriminability of encoded input values.

3. Input Normalization:

- **Effect:** Rate encoding often involves normalizing input data to a common range between 0 and 1.
- **Impact:**
    - Normalization ensures that input data are represented consistently across neurons, facilitating comparability between encoded patterns.
    - Improper normalization or scaling may distort the relationship between input values and firing rates, leading to inaccurate encoding and reduced performance.

4. Encoding Precision:

- **Effect:** The precision of rate encoding determines the granularity of encoded information.
- **Impact:**
    - Higher precision, achieved through fine-tuning rate_min and rate_max, enables neurons to encode subtle variations in input intensity with greater accuracy.
    - Lower precision may sacrifice fine-grained encoding in favor of broader, more general representations, depending on the specific requirements of the task.

1.3 Encoding Using Poisson Distribution

Description: Encoding using Poisson distribution is a method where information is encoded into a sequence of spikes using the Poisson distribution. In this encoding scheme, the firing rate of neurons determines the probability of generating spikes over time.

Implementation Details: The encoding using Poisson distribution was implemented using two main functions: **poisson_spiking** and **rate_encode_input**.

- The **poisson_spiking** function simulates spike generation based on a specified firing rate and simulation time using the Poisson distribution.
- The **rate_encode_input** function encodes a 1D input array into spike trains using rate coding, where firing rates are determined by the input data and normalized between a specified minimum and maximum rate.

Algorithm: The **poisson_spiking** function generates spike times by first calculating the average spike count using the specified firing rate and simulation time. It then samples spike counts from a Poisson distribution and simulates spike times by sampling from an exponential distribution. The **rate_encode_input** function normalizes the input data and calculates firing rates for each neuron based on the input values, which are then used to generate spike trains.

Example Usage and Visualization: An example usage of encoding using Poisson distribution is provided using random input data. The resulting spike trains are visualized using a raster plot, which displays the spike times of neurons over time.

Expected Outcome: Encoding using Poisson distribution is expected to provide a stochastic representation of input data, where the probability of spike generation is determined by the firing rates of neurons. This encoding scheme enables the representation of input data in the form of spike trains, allowing for efficient processing and transmission of information in neural networks

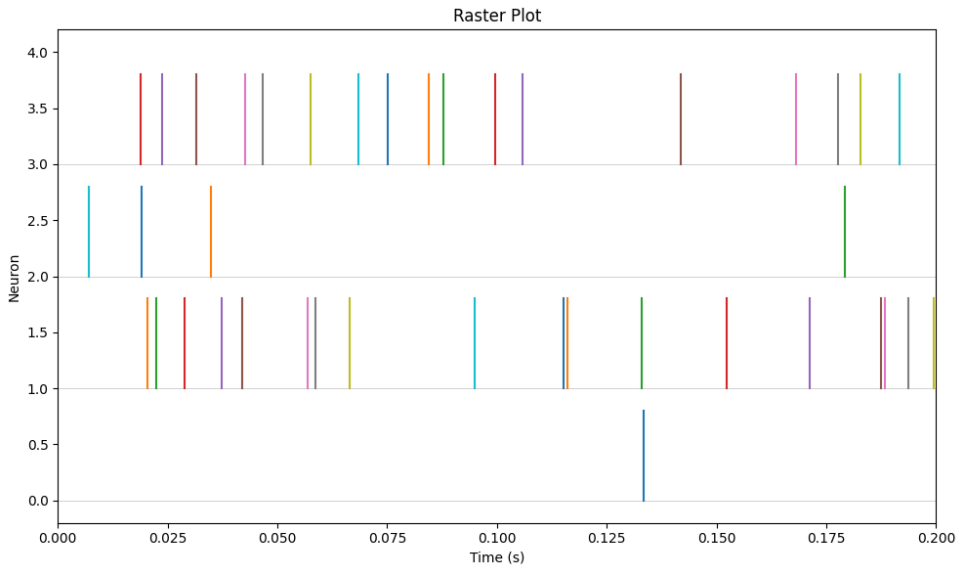


Figure 3 Raster Plot Encoding Using Poisson Distribution

Name: Pooria Heydarian
Student.no: 610399206

Computational Neuroscience

Project three report

Effects of Parameters on Encoding Using Poisson Distribution

1. Firing Rate (rate):

- **Effect:** The firing rate parameter (rate) determines the average rate at which spikes are generated by the neuron.
- **Impact:**
    - Higher firing rates result in more frequent spike occurrences, leading to denser encoding of input information.
    - Lower firing rates produce sparser spike trains, reducing the density of encoded information and potentially affecting the fidelity of representation.

2. Simulation Time (time):

- **Effect:** The simulation time parameter (time) specifies the duration over which spike trains are generated.
- **Impact:**
    - Longer simulation times allow for the accumulation of more spikes, resulting in longer spike trains and potentially richer encoding of input patterns.
    - Shorter simulation times limit the duration of spike trains, constraining the amount of encoded information and potentially leading to lossy representations.

3. Variability in Firing Patterns:

- **Effect:** The Poisson distribution introduces stochastic variability in spike timing, influenced by the firing rate parameter.
- **Impact:**
    - Higher firing rates lead to reduced variability in spike timing, resulting in more regular and predictable firing patterns.
    - Lower firing rates increase variability, introducing randomness and unpredictability into the spike train generation process.

4. Encoding Precision:

- **Effect:** The precision of encoding using Poisson distribution determines the accuracy of representing input information through spike trains.
- **Impact:**
    - Fine-tuning the firing rate parameter allows for precise control over encoding precision, enabling neurons to faithfully represent input stimuli.
    - Adjusting simulation time balances the trade-off between encoding precision and computational efficiency, optimizing the encoding process for specific applications.

Name: Pooria Heydarian
Student.no: 610399206

Computational Neuroscience

Project three report

2. Unsupervised Learning with Spike-Timing-Dependent Plasticity (STDP)

Implementation of unsupervised learning with Spike-Timing-Dependent Plasticity (STDP) involves simulating a neural network model where synaptic connections between neurons are modified based on the relative timing of pre- and post-synaptic spikes. Here's a breakdown of how STDP can be implemented:

1. **Neural Network Setup:**
    - Define the structure of the neural network, including the number of neurons and their connectivity.
    - Specify the dynamics of individual neurons, such as their firing thresholds and refractory periods.
2. **Synaptic Connectivity:**
    - Establish synaptic connections between neurons, either randomly or according to a predefined pattern.
    - Initialize synaptic weights to represent the strength of connections between neurons.
3. **Simulation Loop:**
    - Run the simulation loop for a specified duration of time.
    - During each time step:
        - Update the membrane potential of each neuron based on incoming synaptic currents and external inputs.
        - Check for spike events in pre-synaptic neurons and transmit spikes to post-synaptic neurons accordingly.
        - Record the timing of pre- and post-synaptic spikes for each synaptic connection.
4. **STDP Rule Implementation:**
    - Implement the STDP learning rule within the simulation loop.
    - Calculate the timing difference (Δt) between pre- and post-synaptic spikes for each synaptic connection.
    - Update synaptic weights based on the timing difference:
        - If Δt is positive (pre-before-post), strengthen the synaptic connection (LTP).
        - If Δt is negative (post-before-pre), weaken the synaptic connection (LTD).
        - The magnitude of weight change may depend on factors such as the amplitude of pre- and post-synaptic spikes, learning rates, and the time constants of synaptic plasticity.

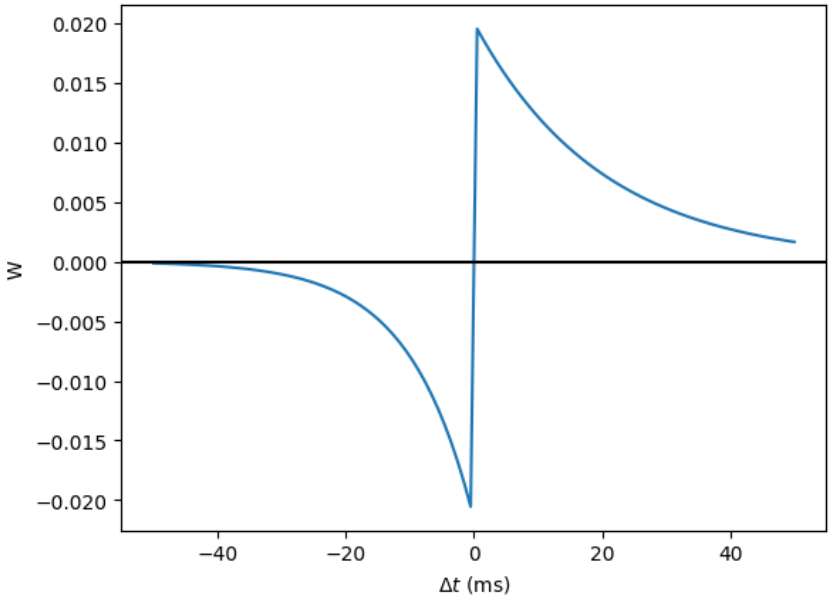
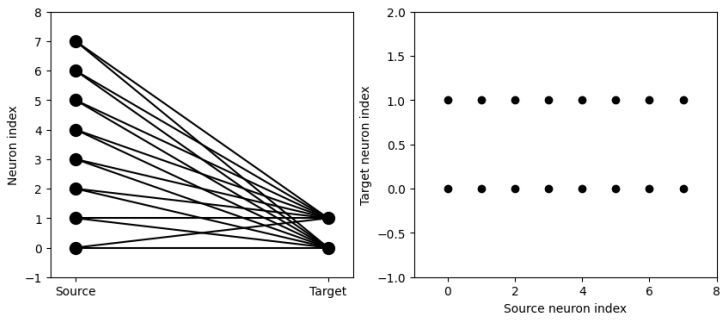


Figure 4 Update synaptic weights based on the timing difference

5. **Visualization and Analysis:**
    - Monitor and visualize the evolution of synaptic weights over time to observe the learning process.
    - Analyze the network's behavior and performance in response to input stimuli to assess the effectiveness of the STDP learning rule.
6. **Parameter Tuning and Optimization:**
    - Fine-tune parameters such as learning rates, time constants, and network architecture to optimize the performance of the neural network.
    - Conduct experiments with different input patterns and stimuli to explore the network's learning capabilities and generalization abilities.

The implementation of Spike-Timing-Dependent Plasticity (STDP) in the project involves simulating a spiking neural network (SNN) consisting of 8 input neurons and 2 output neurons. All input neurons are fully connected to the output neurons, and synaptic connections undergo plasticity according to the STDP learning rule.



Name: Pooria Heydarian
Student.no: 610399206

Computational Neuroscience

Project three report

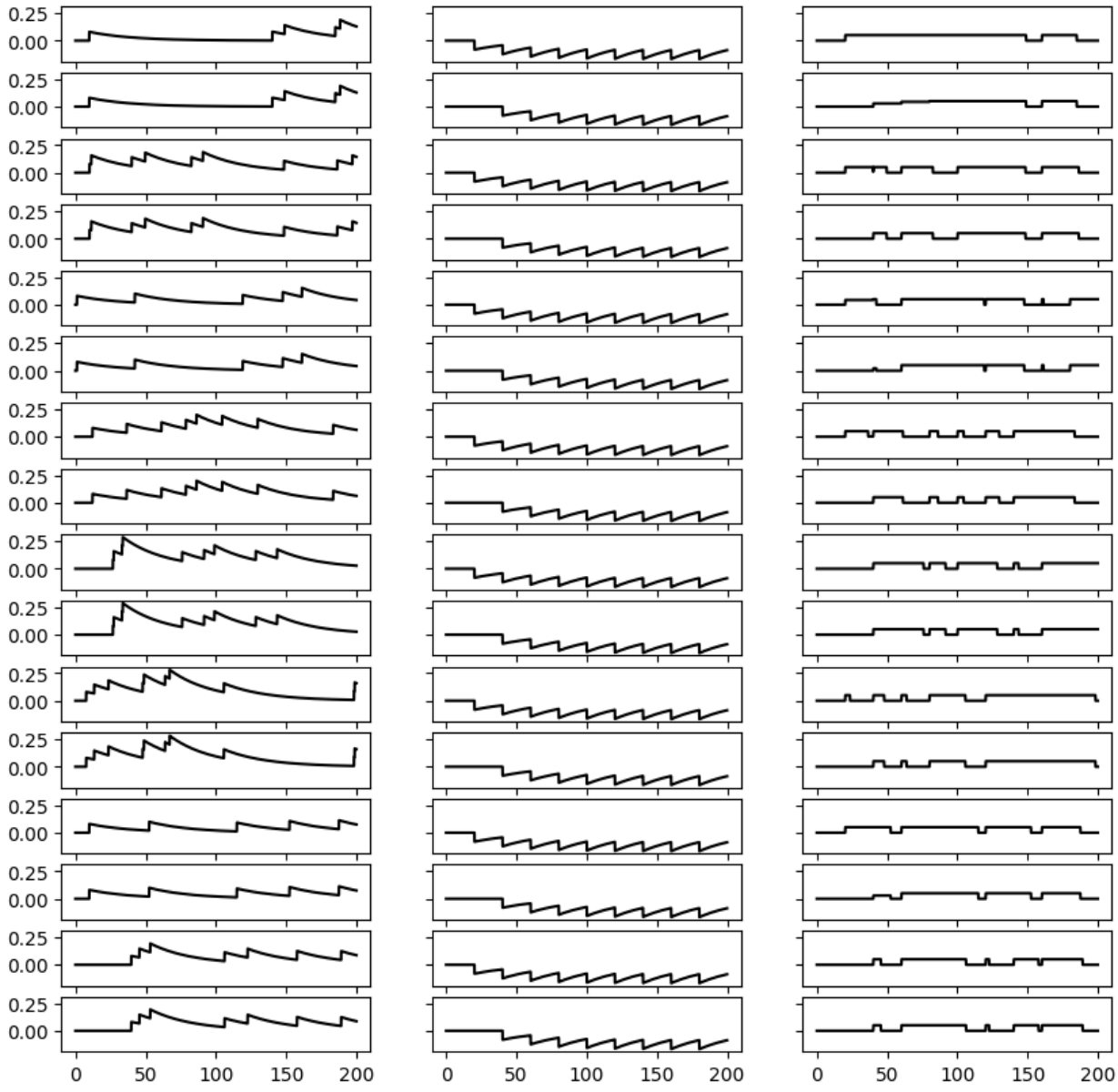


Figure 5 Evolution of STDP Dynamics: In this visual representation, the first column illustrates the variation of A_pre, the second column shows the fluctuation of A_post, and the third column depicts the evolving synaptic weights over learning iterations.

Implementation Details:

- **Network Setup:** The SNN is initialized using the provided code snippet, defining parameters such as the time constants (taupre, taupost), maximum weight (wmax), and learning rates (Apre, Apost).
- **Synaptic Connectivity:** Synaptic connections between the input and output neurons are established using the Synapses object. The connectivity pattern is fully connected, allowing each input neuron to influence both output neurons.

- **STDP Dynamics:** The STDP learning rule is implemented in the Synapses object using clock-driven differential equations for pre- and post-synaptic traces (dapre, dapost).
    - During pre-synaptic events, the synaptic weight is updated based on the post-synaptic trace (apost).
    - During post-synaptic events, the synaptic weight is updated based on the pre-synaptic trace (apre).
    - Weight updates are constrained within the range [0, wmax] to prevent unbounded growth or decay.
- **Visualization and Analysis:** The evolution of synaptic weights, pre-synaptic traces (apre), and post-synaptic traces (apost) are recorded using a StateMonitor object. Visualizations are generated to illustrate the changes in synaptic weights over time.

Assessing Neuronal Similarity using Cosine Similarity

In this section, we leverage cosine similarity as a metric to quantify the similarity between the outputs of the two output neurons in our spiking neural network (SNN). By comparing the firing patterns of these neurons, we gain insights into their learning dynamics and potential convergence towards similar representations.

Methodology:

- **Cosine Similarity Calculation:** Cosine similarity measures the cosine of the angle between two vectors and provides a value between -1 and 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity.
- **Output Encoding:** The firing patterns of the output neurons are encoded as binary vectors, where each element represents the presence or absence of spikes within a specific time window.
- **Similarity Evaluation:** Cosine similarity is computed between the binary vectors representing the outputs of the two neurons across multiple time steps or simulation iterations.

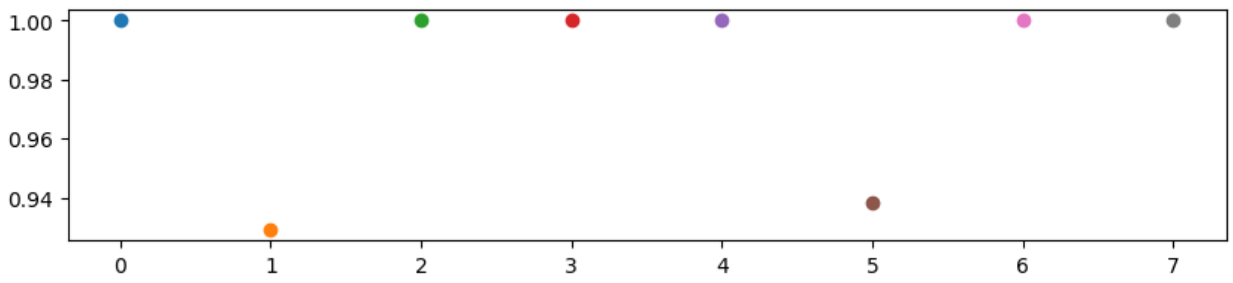


Figure 6 Cosine Similarity

Name: Pooria Heydarian
Student.no: 610399206

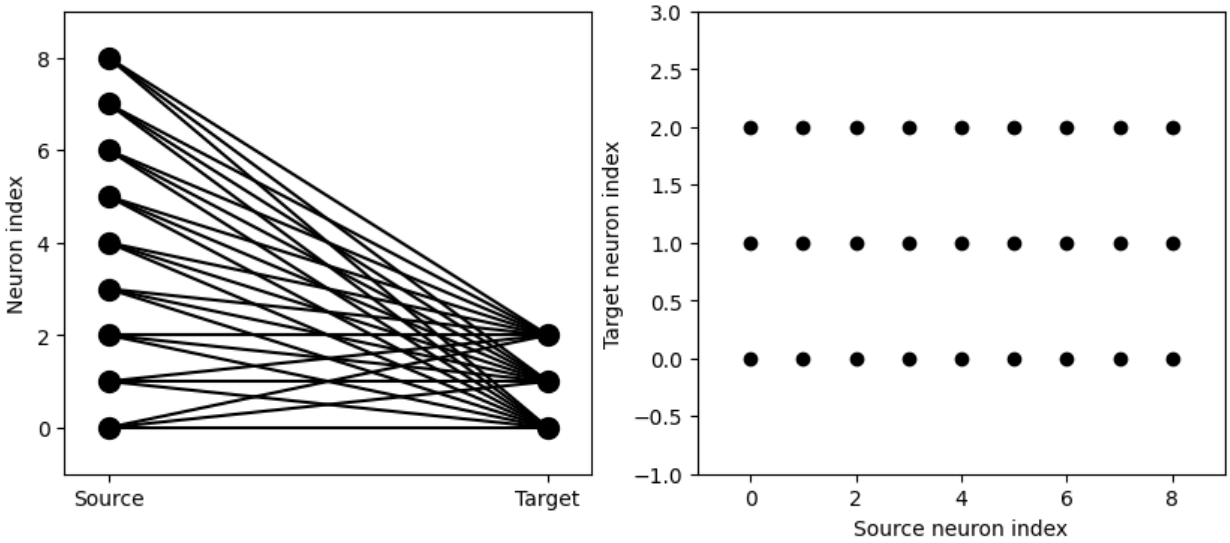
Computational Neuroscience

Project three report

Incorporating Non-Spiking Neurons to Analyze Weight Changes

Overview:

In this section, we introduce non-spiking neurons to each neuron group in our spiking neural network (SNN) to investigate their influence on the changing weight connections during the learning process. By including non-spiking neurons, we aim to explore how the presence of inactive neurons affects synaptic plasticity and network dynamics.



Methodology:

- **Non-Spiking Neuron Integration:** One non-spiking neuron is added to each neuron group, alongside the existing spiking neurons. These non-spiking neurons do not generate spikes during the simulation but contribute to the network's activity through their synaptic connections.
- **Impact on Weight Changes:** The addition of non-spiking neurons introduces additional constraints and dynamics to the learning process. We analyze how the presence of non-spiking neurons influences the changing weight connections between neurons over time.

Name: Pooria Heydarian
Student.no: 610399206

Computational Neuroscience

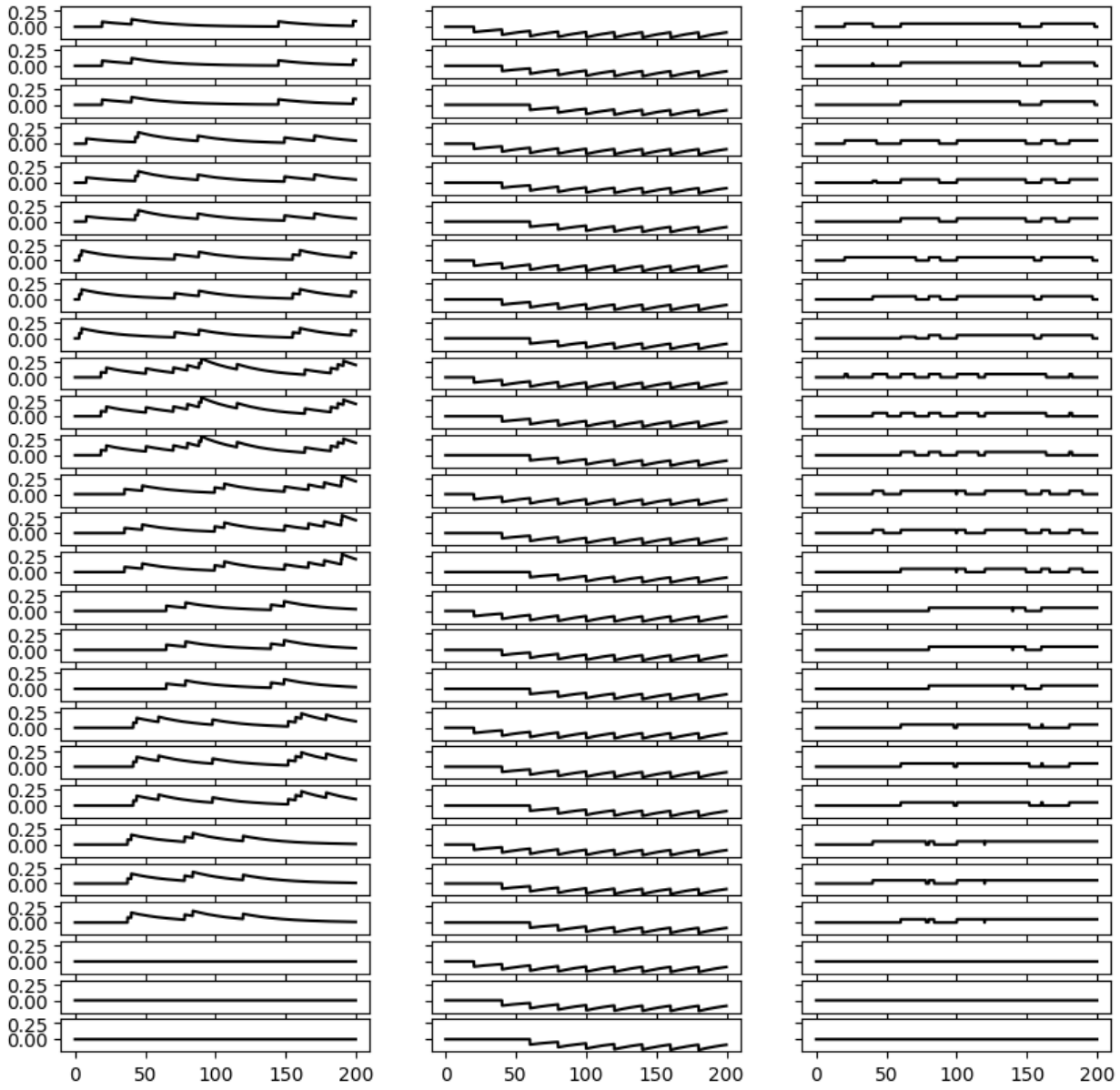


Figure 7 Evolution of STDP Dynamics: In this visual representation, the first column illustrates the variation of Learning Rate pre, the second column shows the fluctuation of Learning Rate post, and the third column depicts the evolving synaptic weights over learning iterations.

Effects of Parameters in Spike-Timing-Dependent Plasticity (STDP)

1. Time Constants (taupre and taupost):

- **Effect:** Time constants determine the integration window for pre- and post-synaptic spikes, influencing the duration over which synaptic changes occur.
- **Impact:**
    - Longer time constants result in a wider integration window, allowing for slower but more persistent changes in synaptic weights.

- Shorter time constants lead to rapid but transient changes, facilitating faster adaptation to recent spiking activity.

2. Maximum Weight (wmax):

- **Effect:** Maximum weight constrains the magnitude of synaptic weight changes, preventing unbounded growth or decay.
- **Impact:**
  - Higher values of wmax allow for greater flexibility in synaptic plasticity, enabling larger weight changes in response to strong spiking correlations.
  - Lower values of wmax limit the extent of synaptic modifications, promoting stability and preventing runaway potentiation or depression.

3. Learning Rates (Apre and Apost):

- **Effect:** Learning rates determine the strength of synaptic modifications during pre- and post-synaptic spike events.
- **Impact:**
  - Higher values of Apre and Apost result in more pronounced synaptic changes, accelerating the learning process and enhancing network plasticity.
  - Lower values of Apre and Apost lead to more gradual weight adjustments, promoting stability and preventing rapid oscillations in synaptic weights.

4. Pre- and Post-Synaptic Traces (dapre and dapost):

- **Effect:** Pre- and post-synaptic traces track the history of pre- and post-synaptic activity, influencing the timing and magnitude of weight changes.
- **Impact:**
  - Increasing values of dapre and dapost amplify the influence of past spiking activity on synaptic plasticity, enhancing the network's ability to learn from temporal patterns.
  - Decreasing values of dapre and dapost reduce the influence of past activity, favoring rapid adaptation to recent spiking events.